Spring 2015

Optimal start-up control of an evaporation system modeled as an interconnected hybrid dynamical system

Rithesh Iyer
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Rithesh Iyer

Entitled
Optimal Start-up Control of an Evaporation System Modeled as an Interconnected Hybrid Dynamical System

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

RAYMOND A. DE CARLO

JIANGHAI HU

ZOLTAN K. NAGY

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): RAYMOND A. DE CARLO

Approved by: Michael R. Melloch 04/27/2015

Head of the Department Graduate Program Date

OPTIMAL START-UP CONTROL OF AN EVAPORATION SYSTEM
MODELED AS AN
INTERCONNECTED HYBRID DYNAMICAL SYSTEM

A Thesis

Submitted to the Faculty

of

Purdue University

by

Rithesh Iyer

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2015

Purdue University

West Lafayette, Indiana

Dedicated to my family and my Alma mater

ACKNOWLEDGEMENTS

In the process of working towards the results of this thesis the guidance of my advisor Prof Raymond DeCarlo has been invaluable. I would specially like to thank Prof. DeCarlo for his patience, willingness and continuous support while guiding me for this work. Without his dedicated effort to work with me on understanding each and every part of the model, it would have been impossible to complete this work in a span of one and a half years. I whole heartedly thank my advisor for his time and efforts.

I would also like to thank the present and past research group members Prof. Rick Meyer, Scott Johnson, Nikhil Jali and Divya Agarwal for the discussions in the process of development of this solver. These discussions were the building blocks of success of the algorithm developed.

In addition to this I would like to thank Prof. C Sonntag for providing important publications related to their work which clarified the relationships used in their work. I would like to thank Prof Zoltan Nagy for encouraging me and guiding me during the meetings with him throughout the process of developing this optimizing solver. I extend my gratitude towards Prof. Carl Liard who motivated me and provided the necessary direction to start this research. In the end I would like to thank my family for their constant support and encouragement to work hard and pursue my interests.

TABLE OF CONTENTS

Page

Page

Page

APPENDICES

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Iyer, Rithesh M.S.E.C.E., Purdue University, May 2015. Optimal start-up control of an evaporation system modeled as an interconnected dynamical system. Major Professor: Raymond DeCarlo.

The purpose of this research is to investigate the feasibility and advantageous outcomes of modeling a complicated non-linear hybrid dynamical process as an interconnected dynamical system for the purpose of solving a hybrid optimal control problem under the framework of nonlinear model predictive control. This work considers a hybrid model of the startup process of an evaporation system. In this evaporation system a liquid containing mixture of a non-volatile component A and volatile solvents B (water) and C (alcohol) is heated to evaporate the solvents and obtain component A at a higher concentration using a column that is temperature controlled by valves that control the flow of steam through a heat exchanger; valves also control the input feed inflow, vapor outflow and the drain of the concentrated product. The hybrid nature of this process was established in the work of C. Sonntag et. al. In this thesis we reformulate the mathematical model as an interconnected dynamical model with two autonomous modes. The reformulated model is then used as a constraint set for the optimization of a performance metric characterizing the startup of the evaporation process and its evolution into steady-state operation. The algorithm used to solve the optimization problem is a new version of the existing algorithms in which the model constraints and cost function computation are performed outside of the sequential quadratic (optimization) program inside $fmincon$ in MATLAB. Extensive comparisons to the work of C. Sonntag et. al. are made.

1. INTRODUCTION

1.1 Introduction

Automated optimal control of chemical units has been a very extensive field of research for the past century and substantial development has been made in the continuous control of chemical processes. Delgass et al. in [20] present a brief history of the evolution of Chemical engineering. The history of chemical process control also traces back to the evolution of batch control of production lines like in the works of G.V. Reklaitis et. al. [17]. This has further advanced into continuous control of MIMO processes, majorly enabled by PID controllers [23], which are extensively used in chemical process control and automation. Work by Z. Nagy et. al. [22],[23], C Cutler et. al. [24], also highlight applications of Advanced Process Control (APC) techniques for process automation.

Any continuous chemical process can be typically divided into transient operation and steady state operation. Steady state operation means that the process variables are controlled in a small neighborhood around the respective set points for the desired process outcomes. Transient operation refers to the case when the process variables undergo major changes in their operating points. This can either happen as a result of unscheduled process disturbances and emergencies or as a result of scheduled start-up and shut-down activities. Safety of operation during start-up is a primary concern for responsible operation of a process. Typically startup of a critical chemical process is performed manually by the plant operators by opening and closing the valves from the control panel to incrementally guide the process into a steady state of operation. The risks involved during startup include environmental impact as a result of process emissions, labor related emergencies due to human error and other emergencies due to process failures. K. Kamarizan et. al.[18] put forth a detailed analysis of the same.

In this context, safe and speedy automated startup of chemical process units with minimum losses is one motivation for the investigation in this thesis. Specifically the focus herein is on a transient startup process of an evaporation system described in [2], in which a liquid mixture containing a low concentration of a non-volatile component and two volatile components is evaporated to obtain a product having higher concentration of the non-volatile component.

1.2 The startup process

The evaporation system typically is operated in multiple stages in which a series of evaporation columns is used to increase the concentration of the product like the system described in [2]. However in this thesis we consider the startup of a single evaporation column as done in [1]. The startup of this process begins with a process liquid mixture entering the evaporation column at 327K containing a non-volatile component A (12%) and volatile solvents B (water, 85%) and C (alcohol, 3%). The liquid standing in the evaporation column is heated by a steam driven heat-exchanger. The temperature of the liquid inside the column rises and the liquid mixture begins evaporating. Thus components B and C are vaporized to increase the concentration of component A in the remaining liquid. The target concentration of component A in the product and the target level of liquid in the column are 80% and 64% respectively. The process can be operated in a steady state operation at these target values.

This target is achieved by controlling the opening and closing of four valves governing fluid inflows and outflows of the process, as described below:

1. The feed inflow valve (V_1) controls the inflow of the process liquid into the evaporation column.
2. The product outflow valve (V_2) controls the outflow of product from the evaporation column bottom.
3. The vapor valve (V_{v1}) controls outflow of evaporated vapor from the column top.
4. The steam valve (V_{v2}) controls the inflow of steam into the heat exchanger for heating the contents of the evaporation column.

Thus the control task in this startup process is to determine the optimal sequence of valve switching/ openings for the above four valves for efficiently achieving the process targets.

In the case of startup of a multi-stage evaporation process in [2] C. Sonntag et. al model all the four valves mentioned above as control valves with continuous inputs. However in the case of startup of a single evaporation column, in [1] the authors model the valves V_{v2} and V_2 as discretely controlled valves. The valve V_{v2} takes two discrete positions: 0 implying 80% opening and 1 implying 100% opening. The valve V_2 takes two discrete positions. 0 implying valve is fully closed (0%) and 1 implying 11.5% opening of the valve [1]. In this thesis, we consider continuous operation of the valves V_{v2} and V_2 in the continuous range of [80%, 100%] and [0, 11.5%] respectively.

The hybrid nature of this process is autonomous due to the change in dynamics of the process when the liquid starts evaporating inside the evaporation column. C Sonntag et. al. [2] have considered this to occur when the pressure above the liquid in the column rises above a value of 0.4 bar. Thus the hybrid switching in this process is state dependent. The mathematical model of an evaporation process operational at the Bayer Chemical Company is detailed in [2]. The process is modeled using a set of Differential Algebraic Equations under a set of assumptions also set forth in [2]. As part of this thesis work we reformulate the same mathematical model as an interconnected dynamical system [15], [16],[36] and solve for optimal startup controls using a different optimization scheme.

1.3 Contents organization

The dynamic model of the startup process established in [1] and [2] is explained in chapter 2. A discussion of the performance index or the cost function formulated for solving the optimization problem is set forth and explained in chapter 3. Chapter 4 covers the reformulation of the model as an Interconnected Dynamical System. Chapter 5 presents a thorough comparison of the results obtained in this work with the results in [1]. The final chapter 6, talks about the limitations of this work and extensions that can be made in the future.

---

[1] One reason for limiting the minimum opening% of the Vv2 to 80% and the maximum opening% of V2 to 11.5% can be to restrict the variation of process variables in a contained range during the startup process using experimental data.

2. DYNAMIC MODEL OF THE EVAPORATION STARTUP

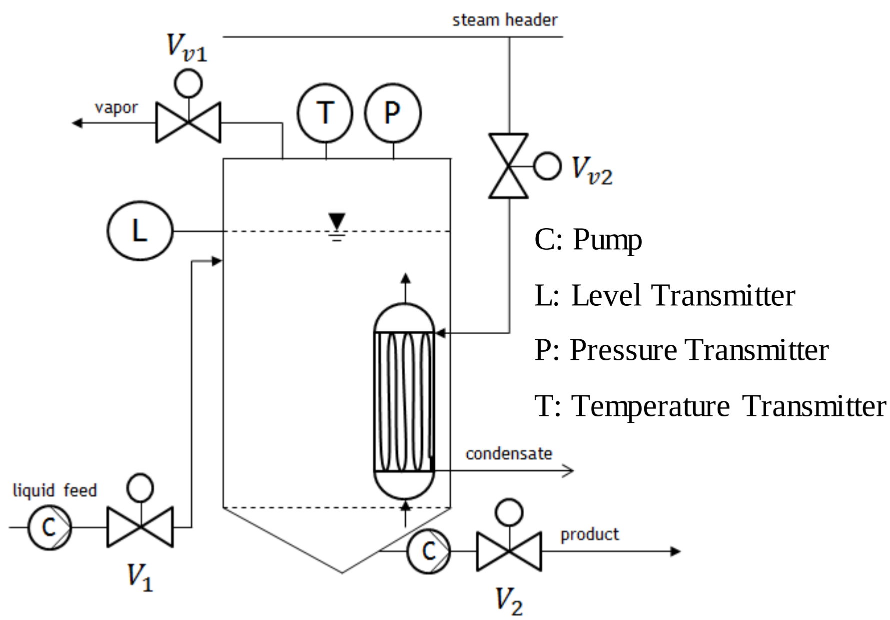


Figure 2.1 Schematic flow diagram of the evaporation process solved for in [1]

The schematic flow diagram of an evaporation system comprised of a single evaporation column is depicted in Figure 2.1. This is identical to the system considered by C. Sonntag et. al. in [1],[2] for solving the startup problem of a single evaporation column. The valves V_1, V_2, V_{v1} and V_{v2} shown in the Figure 2.1 represent the inflow and outflow control valves described earlier in section 1.2. The vessel marked B inside the evaporation column represents the steam driven heat exchanger which heats the liquid contained inside the evaporation column. Symbols containing T, L and P depict the temperature transmitter, level transmitter and pressure transmitters respectively, installed for monitoring the changes in these process variables during the operation of this process. The symbol with C indicates the pumps on the inlet feed line and the outlet product line.

The pumps are used to pump the process liquid into the evaporator and to dispatch the product out of the evaporator. The liquid feed flowing into evaporation column through the inlet valve V_1has three components A, B and C as described earlier in section 1.2. The evaporation system moves between two major modes before reaching the steady state operation. With the process variables at the initial conditions the system starts in the non-evaporating mode and then autonomously switches to an evaporation mode once the pressure over the liquid reaches the value at which the liquid mixture inside the evaporator vaporizes. The process then reaches a steady state at a specific level and concentration of component A in the liquid mixture inside the evaporator. The mathematical model for this process is described in this chapter in the following sections.

2.1 Model equations

In this section we characterize the dynamics of the evaporation column described in the previous section. All the relationships set forth in this work are as per the evaporator model developed by C.Sonntag et.al in [1] and [2]. Section 2.1.1 presents a detailed explanation of the model equations for the Non-Evaporating mode and section 2.1.2 explains the Evaporating mode of the evaporation process.

2.1.1 Non-evaporating mode

As described in section 1.2, during startup of the evaporation process, liquid mixture comprised of components A, B and C enters the evaporation column and is heated. The liquid mixture is heated in the non-evaporating mode until the pressure rises to the value of 0.4 bar [2]. In this section the dynamics of the non-evaporating mode are explained by describing the mass and energy balance equations followed by the algebraic relations between the process variables. The first mass balance equation for component A can be given as:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m}_{o,l} w_A \tag{2.1}$$

where m_A is the mass of component A inside the evaporation column, $\dot{m}_{i,l}$ is input feed flow of the liquid mixture into the evaporation column, $w_{A,i}$ is the mass fraction of component A in the input feed flow in which case $\dot{m}_{i,l} w_{A,i}$ is the mass flow of component A into the evaporation column; $\dot{m}_{o,l}$ denotes the mass out-flow of the liquid exiting the evaporation column while w_A denotes the mass fraction of component A inside the evaporation column which is equal to the mass fraction of component A exiting the evaporation column. Thus we see that the change in mass of component A inside the evaporation column for the non-evaporating mode is precisely what is specified by equation 2.1. Here $\dot{m}_{i,l}$ can be represented as a linear function of the opening of the input valve, V_1, written as,

$$\dot{m}_{i,l} = 10V_1 kg/s \tag{2.1a}$$

and $\dot{m}_{o,l}$ can be represented as a linear function of output valve opening V_2 , written as,

$$\dot{m}_{o,l} = 10V_2 kg/s \tag{2.1b}$$

where the multiplication factor 10kg/s in equation 2.1a and 2.1b scales the opening percentage of the input and output valves respectively expressed in the range of [0,1] to give the rate of mass inflow and outflow respectively. This can be estimated from the input-output experimental data obtained for the valves.

Similarly we can specify the change in mass of components B and C with the same explanation. Thus we have the following equations applicable for components B and C:

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m}_{o,l} w_B \tag{2.2}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m}_{o,l} w_C \tag{2.3}$$

Our 4th differential equation in the non-evaporating mode describes the change in the energy U of the liquid inside the evaporation column. This can be expressed as the difference of energy added through the input flow $\dot{m}_{i,l} c_{p,l}(w) T_i$ and the energy leaving the system $\dot{m}_{o,l} c_{p,l}(w) T$ plus any heat added to the evaporation column denoted by $\dot{Q}$. Hence we have the energy balance equation:

$$\frac{dU}{dt} = \dot{m}_{i,l} c_{p,l}(w) T_i - \dot{m}_{o,l} c_{p,l}(w) T + \dot{Q} \tag{2.4}$$

where $w(t) = [w_A(t), w_B(t), w_C(t)]^T$ is the vector of product mass fractions and $c_{p,l}(w)$is the specific heat of the mixed liquid having mass fraction $w(t)$ that stays constant in the non- evaporating mode. The dynamics of $\dot{Q}$ are described later in section 2.2.

The constraint for mass conservation can be expressed in terms of the mass fractions of liquid components. Since the total mass fraction of components A, B and C is one it follows that:

$$w_A + w_B + w_C = 1 \tag{2.5}$$

Thus the total mass inside the evaporation column is $m_l = w_A m_l + w_B m_l + w_C m_l$ which implies that

$$m_l = m_A + m_B + m_C \tag{2.6}$$

The constraints for distribution of mass of the respective liquid components (m_A, m_B, m_C) can be formulated as the product of the partial fractions of respective components (w_A, w_B, w_C) multiplied by the mass of the liquid contained inside the evaporator (m_l). This gives the following algebraic constraint equations:

$$m_A = w_A\, m_l \tag{2.7a}$$

$$m_B = w_B\, m_l \tag{2.7b}$$

$$m_C = w_C\, m_l \tag{2.7c}$$

Since this is non-evaporation mode, there is no distribution of mass in the vapor phase. So the parameters of mass fraction of component B in vapor phase ξ_B; the mass fraction of component C in vapor phase ξ_C and the volume of vapor inside the evaporator, V_v can be equated to zero.

$$\xi_B = 0 \tag{2.8a}$$

$$\xi_C = 0 \tag{2.8b}$$

$$V_v = 0 \tag{2.8c}$$

The internal energy inside the evaporation column can be given as the product of mass of liquid inside the evaporation column, m_l, the specific heat capacity of the mixed liquid, $c_{p,l}(w)$, and the temperature T inside the evaporation column at that instant. Hence we have the following constraint equation for the internal energy:

$$U = m_l c_{p,l}(w) T \tag{2.9}$$

A derivation of equation 2.4 from equation 2.9 will follow in Appendix-A where we describe the heat exchanger equations. We now proceed to a description of the mixture dynamics in the evaporating mode.

2.1.2 Evaporating mode

In this section the mass balance equation and energy balance equations for the evaporating case are formulated and a development similar to section 2.1.1 is presented. In this overall process, component A remains non-volatile. Hence the mass conservation equation for component A is the same as equation 2.1 repeated below for completeness:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m}_{o,l} w_A \tag{2.10}$$

On the other hand components B and C start evaporating in this mode when the vapor pressure above the liquid surface exceeds the ambient pressure at that temperature. Thus due to this evaporation the terms pertaining to the vapor phase of components B and C must appear in the mass balance equations in addition to the liquid phase terms of equations 2.2 and 2.3. Thus the mass balance equations of components B and C can hence be written as:

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m}_{o,l} w_B - \dot{m}_{o,v}\, \xi_B \tag{2.11}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m}_{o,l} w_C - \dot{m}_{o,v}\, \xi_C \tag{2.12}$$

where m_B, m_C are respectively the masses of components B and C inside the evaporation column, $\dot{m}_{i,l}$ is mass input flow of the liquid into the evaporation column, $w_{B,i}, w_{C,i}$ are respectively the mass fractions of components B and C in the input feed flow in which case $\dot{m}_{i,l} w_{B,i}$ and $\dot{m}_{i,l} w_{C,i}$ are the mass flow of the respective components into the evaporation column; $\dot{m}_{o,l}$ denotes the mass out-flow of the liquid exiting the evaporation column while w_B, w_C denote the mass fractions of components B and C inside the evaporation column which are equal to the mass fractions of component B and C exiting the evaporation column; $\dot{m}_{o,v}$ denotes the mass outflow of the vapor exiting the evaporation column while ξ_B denotes the mass fraction of component B and $\dot{m}_{o,v}\xi_B$ thus gives the mass of component B exiting the evaporation column in vapor phase.

Similar to equation 2.4 in the non-evaporating case, our 4th differential equation 2.14 below for the evaporating mode describes the change in the energy U of the liquid inside the evaporation column. The terms corresponding to the energy of the vapor phase must also appear in the energy balance equation. Thus in this mode the rate of change in energy for the evaporation column system can be described as the energy added to the evaporation column by the input flow, $\dot{m}_{i,l}c_{p,i,l}(w)T_i$ minus the energy of liquid leaving the system $\dot{m}_{o,l}c_{p,l}(w)T$ minus the energy of vapor leaving the evaporation column $\dot{m}_{o,v}\ f_v(T,\xi)$ plus any heat added to the evaporation column denoted by $\dot{Q}$. Here $f_v(T,\xi)$ is a non-linear function modeling the energy content of the vapor exiting the evaporation column and $\dot{m}_{o,v}$ denotes the mass outflow of the vapor exiting the evaporation column. A detailed description of $f_v(T,\xi)$ using thermodynamic models is provided in Appendix-B. Hence we have the following energy balance equation:

$$\frac{dU}{dt} = \dot{m}_{i,l}c_{p,i,l}(w)T_i - \dot{m}_{o,l}c_{p,l}(w).T - \dot{m}_{o,v}\ f_v(T,\xi) + \dot{Q} \tag{2.13}$$

As is done in [1] the mass outflow of vapor $\dot{m}_{o,v}$ can be expressed using a polynomial approximation of Bernoulli's law as:

$$\dot{m}_{o,v} = \left[-0.069\ \sqrt{\rho_v(w,T)}\frac{(P_D^3-36P_D^2-69P_D)}{P_D+0.08}\right]V_{V1} \tag{2.14}$$

where P_D is the difference in pressure inside the evaporation column P and the atmospheric pressure, P_A. Hence we have $P_D = P - P_A$. The term $\rho_v(w,T)$ denotes the density of the vapor phase inside the evaporation column which is a nonlinear function of $w(t) = [w_A(t), w_B(t), w_C(t)]^{T}$ and the temperature T inside the evaporation column. The term V_{V1} denotes the flow rate of vapor through the outlet valve which depends on the external input to the valve.

Similar to equation 2.5, the constraint for mass consistency in the evaporation phase can be expressed in terms of the mass fraction of liquid components and the mass fraction of vapor components. Since the total mass fraction of components A, B and C is one in liquid phase and the total mass fraction of components B and C is one in the vapor phase, it follows that:

$$w_A + w_B + w_C = 1 \text{ and } \xi_B + \xi_C = 1 \tag{2.11}$$

The constraints for distribution of mass of the respective liquid component A remains the same as in equation 2.7a. However the constraint equation for (m_B, m_C) can be formulated as the product of the partial fractions of respective components (w_B, w_C) multiplied by the mass of the liquid contained inside the evaporator (m_l) added with the vapor phase masses of the respective components as described by the right-most terms in the equations 2.16. Specifically,

$$m_A = w_A\, m_l \tag{2.16a}$$

$$m_B = w_B\, m_l + \frac{w_B\, P_B^0(T) V_v}{RT\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]} \tag{2.16b}$$

$$m_C = w_C\, m_l + \frac{w_C\, P_C^0(T) V_v}{RT\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]} \tag{2.16c}$$

where the mass in vapor phase is taken into account for the components B and C through the rightmost terms in the equations 2.16b and 2.16c. $P_B^0(T)$ and $P_C^0(T)$ denote the vapor pressure of components B and C at the evaporation column temperature, T. V_v denotes the volume of vapor phase; M_A, M_B, M_C denote the molecular masses of components A, B and C; R denotes the universal gas constant. In essence,

$$\frac{w_B P_B^0(T) V_v}{RT\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]}$$

gives the mass of component B in the vapor phase derived using the ideal gas law. Equation 2.16c sets forth a similar relationship for component C. It follows that the total mass of the liquid inside the evaporation column for the evaporating phase can be given as:

$$m_l = \rho_l(w)[V - V_v] \tag{2.17}$$

where mass is expressed as the product of density of the liquid inside the evaporation column, $\rho_l(w)$ and the volume of liquid inside the evaporation column which can be expressed as the difference in volume of evaporation column V and the volume of vapor inside the evaporation column V_v. Here $\rho_l(w)$ depends on the mass fraction of individual components $w(t) = [w_A(t), w_B(t), w_C(t)]^{\mathrm{T}}$.

The mass fractions of components B and C in vapor depend on the mass fraction of the respective components in the liquid phase and the partial pressures of the respective components at a particular temperature. This can be written as:

$$\xi_B = \frac{w_B P_B^0(T)}{w_B P_B^0(T) + w_C P_C^0(T)} \tag{2.18a}$$

$$\xi_C = \frac{w_C P_C^0(T)}{w_B P_B^0(T) + w_C P_C^0(T)} \tag{2.18b}$$

where as before w_B, w_C denote the mass fractions of components B and C in liquid phase and $P_B^0(T)$ and $P_C^0(T)$ denote the partial pressure of the respective components.

The internal energy inside the evaporation column for the evaporating mode can be given as the product of mass of liquid inside the evaporation column, m_l , the specific heat capacity of the mixed liquid, $c_{p,l}(w)$, and the temperature T inside the evaporation column at that instant plus the energy of the vapor phase given in the equation 2.10. Thus the internal energy is described by equation 2.19 below:

$$U = m_l c_{p,l}(w)T + \frac{f_v(T,\xi)V_v\left[w_B P_B^0(T) + w_C P_C^0(T)\right]}{RT\left[\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right]} \tag{2.19}$$

where the rightmost term formulates the energy content of the vapor depending on the temperature inside the evaporation column and the partial pressure, volume and partial mass fractions of the components B and C in the vapor phase; recall $f_v(T,\xi)$ as described in equation 2.13 models the energy content of the vapor phase inside the evaporation column. Equation 2.19 can be related to equation 2.13 by extending the derivation in Appendix-A, but such a derivation is beyond the scope of this work.

2.2 Modeling of the heat exchanger

2.2.1 Heat flow in the evaporator

The heat exchanger in this case uses saturated steam as the heating fluid. Inflow of steam is controlled using valve V_{v2} as depicted in Fig1. The heat $\dot{Q}$ flowing from the heat exchanger into the evaporation column can be expressed in terms of the heat exchange coefficient k; the surface area of the heat exchanger, A, and the difference in temperature of the heat exchanger surface and the evaporation column. Thus we conclude:

$$\dot{Q} = kA(T_{HE} - T) \tag{2.20}$$

where, as before T is the temperature of the fluid in the column, and T_{HE} is the temperature of the surface of the heat exchanger column. This is generally same as the temperature of the condensed steam in the heat exchanger. In this process fresh steam enters into the heat exchanger at 440K and 5bar pressure and gets condensed. The heat lost by the steam in this condensation process is transferred to the evaporator to heat the liquid mixture inside the evaporation column.

2.2.2 Relationship to steam inflow

Considering the loss of heat from steam to be fully converted into the heat transferred to the evaporation column the following equation holds for the heat inflow $\dot{Q}$:

$$\dot{Q} = \left[c_{p,v,i}T_{is} - c_{p,l,i}T_{HE} + h_{V,i}\right]\dot{m}_{HE} \quad (2.21)$$

where $c_{p,v,i}$is specific heat capacity of the steam flowing into the exchanger, T_{is} is the steam temperature, $c_{p,l,i}$ is specific heat capacity of liquid flowing out from the exchanger and $h_{V,i}$ denotes enthalpy of vaporization of the steam. The product $c_{p,v,i}.T_{is}$ hence represents the energy content of the steam and $c_{p,l,i}.T_{HE}$ represents the energy content of the condensed liquid flowing out of the exchanger. To arrive at equation 2.21 we need to multiply the sum by the mass of steam flowing into the exchanger i.e. $\dot{m}_{HE}$.

For control purpose we relate $\dot{m}_{HE}$ with the input steam valve opening percentage V_{V2} using polynomial approximation of Bernoulli's law as:

$$\dot{m}_{HE} = -0.069\,\sqrt{\rho_{v,i}\frac{\left(P_{D,i}^{3}-36P_{D,i}^{2}-69P_{D,i}\right)}{P_{D,i}+0.08}}\;V_{v2} \quad (2.22)$$

where $P_{D,i} = P_i - P_{HE}$ i.e. the difference of pressure of the incoming steam P_i and the pressure inside the heat exchanger P_{HE}; $\rho_{v,i}$ denotes the density of the steam flowing into the heat exchanger. Further, using ideal gas law the pressures P_{HE} and P_i can be related by the following equation.

$$P_{HE} = \frac{T_{HE}}{T_{is}}P_i \quad (2.23)$$

This follows from the property that for a given volume of steam, the ratio of pressure and temperature inside the heat exchanger is equal to the ratio of pressure and temperature of the incoming steam.

2.3 Output variables

The measured variables in this process are the temperature, pressure and the level of liquid inside the evaporation column. The pressure inside the evaporation column varies depending on the evaporation of the solvent components while the level varies as a function of the flow of the liquid entering into the evaporation column relative to that flowing out. Temperature in turn depends on the pressure.

The output function is hence defined as the vector of pressure of vapor inside the evaporation column, P and the level of liquid inside the evaporation column, L. Following equations 2.24 and 2.25 describe the P and L

$$P = \frac{\left[\frac{w_B}{M_B}.P_B^0(T)+\frac{w_C}{M_C}.P_C^0(T)\right]}{\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]} \tag{2.24}$$

where P the pressure inside the evaporation column is defined as the sum of partial pressures of components B and C expressed in terms of the partial pressures $P_B^0(T), P_C^0(T)$ and mass fractions $(w_{A,}w_B, w_C)$ of components A, B and C.

The level of liquid in the evaporation column can be written as:

$$L = \frac{\frac{m_l}{\rho_l(w)} - V_B}{v_L} \tag{2.25}$$

where v_L and V_B are structural parameters of the evaporation column. Intuitively the level is calculated as the volume of liquid inside the evaporation column $(\frac{m_l}{\rho_l(w)})$ divided by the cross sectional area of the evaporation column (v_L). Given these relationships, the output function is defined as $y = [L, P]$

2.4 Mode transition

After enlisting the model equations for the non-evaporating and the evaporating modes, in this section we define the conditions governing the movement of the system from one mode to the other mode. The mode transition from the non-evaporating mode to the evaporating mode and vice versa as described in [2] occurs when the pressure inside the evaporation column crosses the value of $P = 0.4\,bar$. Conversely, the reverse happens when pressure inside the column drops below this pressure value.

2.5 Summarizing the model equations

In this section we summarize the model equations discussed in this chapter to describe a complete model of the evaporation system.

2.5.1 Non-evaporating mode

Here we list the four underlying differential equations for the non-evaporating mode. Three of the differential state equations are for mass fractions of the liquid components. The fourth differential equation represents an energy balance as discussed in section 2.1.1:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m}_{o,l} w_A \qquad \text{(from 2.1)}$$

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m}_{o,l} w_B \qquad \text{(from 2.2)}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m}_{o,l} w_C \qquad \text{(from 2.3)}$$

$$\frac{dU}{dt} = \dot{m}_{i,l} c_{p,i,l}(w) T_i - \dot{m}_{o,l} c_{p,l}(w) T + \dot{Q} \qquad \text{(from 2.4)}$$

With the mass inflow and outflow given as:

$$\dot{m}_{i,l} = 10 V_1 \text{ kg/s} \qquad \text{(from 2.1a)}$$

$$\dot{m}_{o,l} = 10 V_2 \text{kg/s} \qquad \text{(from 2.1b)}$$

The thermodynamic relationships for functions $c_{p,i,l}(w)$ and $c_{p,l}(w)$ used in the equations in this section are defined in Appendix-B. Relation equations for other variables are as given below. Consistency of mass fractions can be represented by the following equation:

$$w_A + w_B + w_C = 1 \qquad \text{(from 2.5)}$$

Mass of each component inside the evaporator is given as:

$$m_A = w_A \, m_l \qquad \text{(from 2.7a)}$$

$$m_B = w_B \, m_l \qquad \text{(from 2.7b)}$$

$$m_C = w_C \, m_l \qquad \text{(from 2.7c)}$$

Due to absence of vapor state in the non-evaporative mode the mass fractions of the volatile components and the volume of vapor in the evaporator can be set to zero.

$$\xi_B = 0 \qquad \text{(from 2.8a)}$$

$$\xi_C = 0 \qquad \text{(from 2.8b)}$$

$$V_v = 0 \quad \text{(from 2.8c)}$$

The total energy within the evaporator is then formulated as:

$$U = m_l c_{p,l}(w)T \quad \text{(from 2.9)}$$

where $c_{p,l}(w)$ is the specific heat capacity of the liquid modeled as a linear function of the mass fractions of components A, B and C as described in Appendix B.

2.5.2 Evaporating mode

The four differential equations representing the dynamics of the state variables in the evaporative phase are given as follows:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m}_{o,l} w_A \quad \text{(from 2.10)}$$

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m}_{o,l} w_B - \dot{m}_{o,v}\, \xi_B \quad \text{(from 2.11)}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m}_{o,l} w_C - \dot{m}_{o,v}\, \xi_C \quad \text{(from 2.12)}$$

$$\frac{dU}{dt} = \dot{m}_{i,l} c_{p,i,l}(w) T_i - \dot{m}_{o,l} c_{p,l}(w) T - \dot{m}_{o,v}\, f_v(T,\xi) + \dot{Q} \quad \text{(from 2.13)}$$

where $f_v(T,\xi)$ represents the heat content in vapor given as a non-linear function of temperature and composition ξ in vapor. The thermodynamic relationship for $f_v(T,\xi)$ is provided in Appendix-B. Also note that in addition to the liquid components the mass balance equations 2.11 and 2.12, now include terms with vapor mass fractions. The relation equations of the other evaporator variables are as listed below.

Mass flow rate of vapor state is:

$$\dot{m}_{ov} = \left[-0.069\ \sqrt{\rho_v(w,T)}\, \frac{(P_D^3 - 36P_D^2 - 69P_D)}{P_D + 0.08}\right] V_{V1} \quad \text{(from 2.14)}$$

where, $\rho_v(w,T)$ represents density of vapor and P_D=$P - P_A$. Consistency of the mass fractions is as before with the addition of the vapor mass fraction of components B and C.

$$w_A + w_B + w_C = 1 \text{ and } \xi_B + \xi_C = 1 \quad \text{(from 2.15)}$$

The distribution of mass in vapor and liquid phases is:

$$m_A = m_l w_A \quad \text{(from 2.16a)}$$

$$m_B = m_l w_B - \frac{w_B\, P_B^0(T) V_v}{RT\left[\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right]} \quad \text{(from 2.16b)}$$

$$m_C = m_l w_C - \frac{w_C\, P_C^0(T) V_v}{RT\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]} \quad \text{(from 2.16c)}$$

$$m_l = \rho_l(w)[V - V_v] \quad \text{(from 2.17)}$$

The mass fraction of vapor products is given as:

$$\xi_B = \frac{w_B P_B^0(T)}{w_B P_B^0(T)+w_C P_C^0(T)} \quad \text{(from 2.18a)}$$

$$\xi_C = \frac{w_C P_C^0(T)}{w_B P_B^0(T)+w_C P_C^0(T)} \quad \text{(from 2.18b)}$$

Finally, the total energy within evaporator is given as the sum of the heat content of the liquid state and that of the vapor state:

$$U = m_l c_{p,l}(w)T + \frac{f_v(T,\xi) V_v\left[w_B P_B^0(T)+w_C P_C^0(T)\right]}{R.T\left[\frac{w_A}{M_A}+\frac{w_B}{M_B}+\frac{w_C}{M_C}\right]} \quad \text{(from 2.19)}$$

2.5.3 Heat exchanger model

The equations for the heat flowing from the heat exchanger to the evaporator is:

$$\dot{Q} = kA(T_{HE} - T) \quad \text{(from 2.20)}$$

where K and A are structural parameters of the heat exchanger.

Heat transferred from steam to the heat exchanger is modeled as:

$$\dot{Q} = \left[c_{p,v,i} T_{is} - c_{p,l,i} T_{HE} + h_{V,i}\right] \dot{m_{HE}} \quad \text{(from 2.21)}$$

where $h_{V,i}$is a fixed structural parameter of the heat exchanger. The steam flow equation through the valve V_{v2} is:

$$\dot{m_{HE}} = -0.069\sqrt{\rho_{v,i}}\,\frac{\left(P_{D,i}^3 - 36P_{D,i}^2 - 69P_{D,i}\right)}{P_{D,i}+0.08} V_{V2} \quad \text{(from 2.22)}$$

where $\rho_{v,i}$ is the vapor density of steam. The pressure difference is modeled as:

$$P_{D,i} = P_i - P_{HE}$$

where using ideal gas law:

$$P_{HE} = \frac{T_{HE}}{T_i} P_i \quad \text{(from 2.23)}$$

2.5.4 Output variables

The level of liquid inside the evaporation vessel and the pressure of gases inside the evaporator are the output functions given as follows:

$$P = \frac{\left[\frac{w_B}{M_B} P_B^0(T) + \frac{w_C}{M_C} P_C^0(T)\right]}{\left[\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right]} \qquad \text{(from 2.24)}$$

$$L = \frac{\frac{m_l}{\rho_l(w)} - V_B}{v_L} \qquad \text{(from 2.25)}$$

Table 2.1 Variables and constants

Symbol	**Parameter**	**Value**	**Units**
$\dot{m}_{i,l}$	Mass inflow of feed liquid mixture	Input dependent	kg/s
$\dot{m}_{o,l}$	Mass outflow of product	Input dependent	kg/s
$\dot{m}_{o,v}$	Mass flow of vapor from the evaporator	Input dependent	kg/s
m_l	Mass of liquid inside evaporation column	Variable	kg
w_A	Mass fraction of A in liquid	Variable	$mass\ ratio$
w_B	Mass fraction of B in liquid	Variable	$mass\ ratio$
w_c	Mass fraction of C in liquid	Variable	$mass\ ratio$
w_{Ai}	Mass fraction of A in feed inflow	0.12	$mass\ ratio$
w_{Bi}	Mass fraction of B in feed inflow	0.85	$mass\ ratio$
w_{Ci}	Mass fraction of C in feed inflow	0.03	$mass\ ratio$
ξ_B	Mass fraction of B in vapor	Variable	$mass\ ratio$
ξ_C	Mass fraction of C in vapor	Variable	$mass\ ratio$
α, β	Weights in the cost function	$-$	
$\dot{m}_{HE}$	Steam mass flow rate into heat exchanger	Input dependent	kg/s
L	Level of liquid in evaporator	Variable	$\%$
T	Temperature of liquid in evaporator	Variable	K
ρ_l	Density of liquid in evaporator	Variable	Kg/m^3
ρ_v	Density of vapor in evaporator	Variable	Kg/m^3
V_v	Volume of vapor inside the evaporator	Variable	m^3
$\rho_{v,i}$	Density of steam at P_i, T_{is}	3	Kg/m^3
$c_{p,l,i}$	Specific heat capacity of condensed steam	4.18	KJ/KgK
$c_{p,v,i}$	Specific heat capacity of steam at P_i, T_{is}	2.31625	KJ/KgK
P_i	Pressure of steam input	5	bar
T_{is}	Temperature of steam input	440	K
T_{HE}	Temperature of heat exchanger	Variable	K
P_{HE}	Pressure inside heat exchanger	Variable	bar
k	Heat transfer coefficient at heat ex.	4	KW/m^2K
A	Area of heat ex. Surface	300	m^2
$\dot{Q}$	Heat flow from heat ex. to evaporator	Variable	KJ/s
V_1	Input to feed line valve (%opening)	Input variable	$\%(0-100)$
V_2	Input to product line valve (%opening)	Input variable	$\%(0-11.5)$
Vv_1	Input to vapor outlet valve(%opening)	Input variable	$\%(0-100)$
Vv_2	Input to steam valve(%opening)	Input variable	$\%(80-100)$
v_L	Design parameter of evaporation column	0.058	$m^3/\%$
V_B	Volume of liquid in column at $L = 0\%$	10	m^3
R	Universal gas constant	0.08314472	$\frac{bar.m^3}{kmol.K}$

3. FORMULATING THE PERFORMANCE INDEX

3.1 Introduction and background

"Economic optimization of the process is one of the most important requirements of commercial production of industrial chemicals" [10]. In industrial processes economic optimization is most commonly attained by predictive control methods or by dynamic optimization methods. Optimal control is a wide branch of control theory which has been commonly used for optimization of chemical processes for over a decade [1], [10]. Real-time optimization and fully automated computer control of chemical processes is a relatively recent technology.

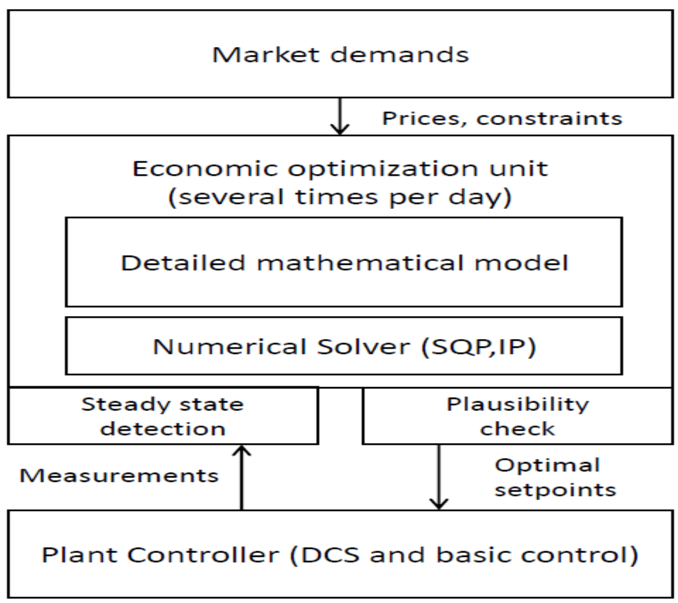


Figure 3.1 Block diagram of computer based online optimization of a chemical process unit [10]

Figure 3.1 depicts the block diagram representing computer based online optimization of a chemical process unit for profit described in [10]. The lowest level of block-diagram represents the process plant operation using the plant level Distributed Control System (DCS) [10]. The plant controller receives set-points for its process variables for optimal operation based on an economic optimization unit. The set-points are checked for plausibility before feeding them into the plant controllers in order to avoid errors and plant upsets. The economic optimization unit uses a detailed mathematical model of the chemical process to solve for the optimal set-points. The numerical solver can be based on one of the conventionally used optimization algorithms like sequential quadratic programing (SQP) [28], interior point algorithm [29] etc. The economic optimization shown in this figure takes into account market demands and current plant measurements (made at steady-state operation) as inputs for computing the optimal set-points. The optimization approach is generally based on an objective function or a performance metric which is the primary focus in this chapter. Figure 3.2 gives a contour line plot of an economic objective function depending on two free constrained variables; as depicted in [10]. In this section we discuss the optimal control and the performance metrics of the hybrid evaporator process modeled in Chapter 2.

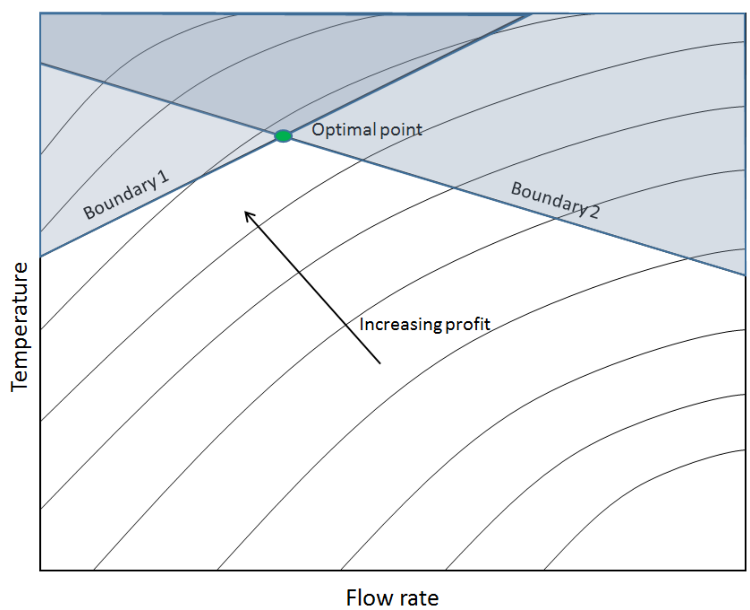


Figure 3.2 Contour line plots of an economic objective function depending on two constrained variables [10]

3.2 Optimal control theory

The history of optimal control theory dates back to 1697 and started with attempts to solve the "Brachystochrone Problem" of finding the fastest trajectory between two spatially located points. Methods based on Newton's calculus of variation (1685) were known earlier. However computational methods like the Euler-Lagrange control equations (1755) [30] and Hamiltonian based computation of optimal control inputs over the state trajectory (1865) [26] and Pontryagin's maximum principle of optimal control theory (1956) [27] represent significant milestones in the history of optimal control theory.

Through the development of computational models and computer based simulations, methods like model predictive control (1976) [25], dynamic matrix control [2](1979) [24], optimal control of mixed logical dynamical systems [9] and gradient based optimal control techniques [12] were made possible. However, there is much ongoing research for modeling and optimal control of continuous and hybrid systems. Most methods involve simulation of the model for state trajectories as part of a control decision to minimize an objective function also termed the performance index (PI). A PI can be used to measure the optimal behavior of a chemical process. A typical PI is a suitable cost function based on the process requirements and objectives. It often involves some measure of efficiency to reduce the process cost. We discuss the qualitative basis for formulating the cost function for this evaporation process in the following sections.

3.3 Attributes of the performance index

In industrial applications, the overall PI can be measured in terms of the cost incurred during the life cycle of the process. Broadly speaking, this can be segregated into the operating cost, maintenance expenditures and the development cost. Startup and shutdown of any chemical process typically leads to losses due to production down-time and potentially lost sales. On the other hand the PI is also impacted by accidents related

---

[2] Method developed by Prof. Charles Cutler. Implemented in MaxAPC-leading Advanced Process Control suite incorporated by Shell Oil and Gas and other Middle Eastern Oil and Gas firms

to labor, environment and the process itself. Accidents occur quite frequently during startup and shutdown [20]. In this work our PI is mainly focused on monetary gains by reducing the down-time, maintaining the state variables close to target values and reducing the consumption of energy resources during the startup process of the evaporator. Although safety can be incorporated into the PI, such is beyond the scope of this work.

The initial primary goal for optimized startup control of the evaporation process is to reach a stable steady state operation of the process with state variables tracking the target values as quickly as possible while minimizing consumption of energy in the form of heat transferred from the steam exchanger. At the end of the startup process we intend to have a continuous steady state process operating for producing component A at the target concentration. Intuitively speaking, the cost function formulated for optimal startup control of the process must essentially include terms corresponding to minimization of deviation of the state trajectories from target values and an energy minimization term. This will ensure that state variables reach the target values as quickly as possible. Here we also observe that there is a trade off in minimizing both tracking deviation and energy consumption. For example, forcing state variables to track the set-points faster will typically consume a higher amount of energy.

Thus in order to account for the above requirements, the basis for formulating the cost function for this evaporation process can be listed as follows:

i. Startup being a transient process, the initial goal is to evolve the states to their target values as efficiently as possible by minimize a transient tracking error.
ii. Maintaining the minimum deviation of state trajectories from the target states once steady state operation is achieved.
iii. Minimization of energy consumption, which includes the heat requirement for the startup process and the cost of the associated inputs.

Hence we define a generalized cost function to satisfy the above requirements denoted as a sum of the costs for state deviation, J_S and cost for energy minimization, J_E:

$$J = J_S + J_E \tag{3.1}$$

In the next sections we discuss the cost function considered in other references for optimal control of the evaporation process. This is followed by a description of an intuitive understanding of the cost for the evaporation process. In the final section of this chapter we arrive at an appropriate cost function formulation for the evaporation process.

3.3.1 Analysis of the cost function formulation

For meeting the requirements of the PI as described in the previous section, the cost function can be a combination of the deviation of intermediate states from the target set plus the energy consumption evaluated in terms of the amount of steam consumed in the process. As discussed in [5] the latter part of this cost function can be formulated in terms of energy usage measured in terms of mass units of saturated steam used for evaporating one unit mass of the solvent summed with the square of deviation of state from the target set. In our case this can be formulated as:

$$J_S = \int_{T_0}^{T_f} \left(\hat{x}_f - x(t)\right)^2 \tag{3.1a}$$

$$J_E = \frac{\int_{T_0}^{T_f} \dot{m}_{HE}.dt}{\int_{T_0}^{T_f} (\dot{m}_{i,l} - \dot{m}_{o,l}).dt} \tag{3.1b}$$

In regards to J_E,

$$\int_{T_0}^{T_f} \dot{m}_{HE}.dt \tag{3.1c}$$

is the mass of steam inflow over $[T_0, T_f]$ which is a function of steam inflow valve (input V_{v2}). This quantity is divided by the mass of solvent evaporated which is given by the integral of the difference in mass inflow and outflow of liquid in the evaporation column; i.e.,

$$\int_{T_0}^{T_f} (\dot{m}_{i,l} - \dot{m}_{o,l}).dt \tag{3.1d}$$

which depends on the control of feed inflow valve (input V_1) and product outflow valve (input V_2). The ratio of (3.1c) and (3.1d) represents the amount of steam consumed per unit mass of the solvent evaporated which measures energy consumed and thus the energy cost. On the other hand J_s is the integral of the squared error in state tracking over $[T_0, T_f]$. Minimization of this cost function forces the reduction of the area under the squared error and is generally achieved in near minimum time depending on other trade-offs in an optimization.

As discussed in [4] the cost function can alternatively be formulated as the summation of deviation of discrete states from the reference values and the cost associated with inputs. Specifically:

$$J_S = \sum_{k=0}^{N} \mu_1 (\hat{\mathrm{x}}_{kf} - \mathrm{x}_{\mathrm{k}}(t_k))^2 \tag{3.2a}$$

$$J_E = \mu_2 \sum_{k=0}^{N} (\mathrm{u}_{\mathrm{k}}(t_k))^2 \tag{3.2b}$$

where the RHS in 3.2a is the Euler discretized form of equation 3.1a and N represents the number of samples over $[T_0, T_f]$. J_E in equation (3.2b) represents the sum of squares of the inputs given to the system multiplied by an associated weight μ_2. The inputs u_k in [4] are taken as the control moves applied to the system for solving a generalized predictive control problem [13]. We interpret them as the discrete sequence of inputs given to the valves for optimal control of the evaporation process described in [4]. In our case we can map them to the controls governing the operation of four valves in the evaporation process (V_1, V_2, V_{v1} and V_{v2}) as described in section 1.2. Hence, it can be seen that both the equations (3.1b) and (3.2b) are essentially dependent on the control inputs. In equation 3.2b, J_E is formulated as the weighted sum of squared control inputs, and in equation 3.1b, J_E is formulated as a ratio of the integral of control inputs over time.

Moreover from the above discussion we see that the choice of the cost function is based on the optimal control methodology, control inputs and the deviation of states from the process targets. In the next section we present an intuitive understanding of the cost function as discussed in [1] and arrive at the formulation of the cost function for this work in the final section of this chapter.

3.3.2 Intuitive understanding of the cost function

For the evaporation process described in this work we can intuitively state that: “the fastest way to increase the component A concentration is to maximize the energy transfer rate $\dot{Q}$ from the heat exchanger to the evaporation vessel” [1], which in turn leads to faster evaporation of the volatile components. From equation 2.20 we see that this energy transfer $\dot{Q}$ is proportional to the difference in temperature of the heat exchanger surface and temperature of the liquid inside the evaporation column. This intuitively suggests that in order to maximize $\dot{Q}$ the temperature of the liquid in the evaporation column needs to be reduced. We observe that the inflowing liquid is at a lower temperature than the heated liquid contained inside the evaporation column. Hence more inflow of the liquid from the feed line will reduce the temperature inside the evaporation column. Thus if the level inside the evaporator column is increased by allowing more inflow of the liquid through the inflow valve V_1, this will in effect reduce the temperature inside the column. It follows intuitively that the liquid level inside the column must be maximized so that temperature of the liquid inside the column is minimized. This creates the maximum temperature gradient between the heat exchanger surface and the liquid in the evaporation column thus forcing maximum heat transfer. Once the target concentration of component A is reached, the next target is to maintain the product concentration and to operate the process at the target liquid level. Thus subsequently the liquid level must be driven to the target level or the maximum attainable safe level inside the evaporation column.

Thus we see that the cost function in [1] denoting the instantaneous cost is defined as a function of the concentration of the component A, w_A, and the level of liquid in the evaporation column L. It is expressed by the heuristic formulation:

$$J_{inst} = \begin{cases} \alpha_1 |w_{s,A}(t) - w_{s,A,target}| + \beta_1 |L_s(t) - L_{s,\max}| & if\ |w_{s,A}(t) - w_{s,A,target}| > 0.09 \\ \alpha_2 |w_{s,A}(t) - w_{s,A,target}| + \beta_2 |L_s(t) - L_{s,\text{target}}| & if\ |w_{s,A}(t) - w_{s,A,target}| \le 0.09 \end{cases} \tag{3.3}$$

where, $w_{s,A}$ represents the scaled concentration of component A, $w_{s,A,target}$ represents the target concentration of component A, L_s denotes the scaled level inside the evaporation column, $L_{s,\text{target}}$ denotes the target level inside the evaporation column at steady state and $L_{s,\max}$ denotes the safe maximum level inside the evaporation column scaled to 1; thus $L_{s,\max}$ =1. It is to be noted that the values of the variables in the cost function are scaled to lie in the interval [0,1]. For example suppose the maximum level of the liquid in the evaporation column is $L_{\max} = 90\%$, and the level of the liquid inside the column column is $L = 30\%$. The normalized values $L_{s,\max}$ and L_s will be calculated as 1 [meaning 90%] and 0.27 [= 0.3x0.9] respectively. Such a scaling is considered necessary so that the values of the level and concentration have comparable impact on the cost function. The parameters α_i and β_i are the weights based on the requirements of the process in the respective modes. The ratio $\frac{\alpha_1}{\beta_1}$ is set to 4 when $|w_{s,A} - w_{s,A,target}| > 0.09$ to give more weight on state tracking. This ensures that w_A increases towards the target value. On the other hand when $|w_{s,A} - w_{s,A,target}| \leq 0.09$ the goal of the controller is to track the steady-state target level inside the evaporation column. Hence $\frac{\beta_2}{\alpha_2}$ is set to 4 to assign higher weightage to level tracking.

In effect the integration of the instantaneous cost over the interval $[T_0, T_f]$ is quite similar to our definition of cost as:

$$J_S = \int_{T_0}^{T_f} J_{S_{inst}}\, dt = \int_{T_0}^{T_f} (\alpha_i |w_{s,A}(t) - w_{s,A,target}|)\, dt \tag{3.3a}$$

$$J_E = \int_{T_0}^{T_f} J_{E_{inst}} dt = \int_{T_0}^{T_f} (\, \beta_i |L_s(t) - L_{s,\text{target}}|\,)\, dt \tag{3.3b}$$

Here the integrand $J_{S_{inst}}$ denotes the instantaneous cost of deviation of the states from target values and $J_{E_{inst}}$ denotes the instantaneous cost of deviation of level from the target values. Thus 3.3a represents the cost associated with the deviation of scaled concentration of component A from scaled target concentration of component A integrated over the interval $[T_0, T_f]$ and 3.3b represents the cost associated with the deviation of the scaled level inside the evaporation column from scaled target level, over the interval $[T_0, T_f]$. In the next section we formulate an integral quadratic cost function for this work similar to the cost function discussed in this section.

3.3.3 Formulating an integral quadratic cost function

Following the intuitive understanding of the cost function in the preceding sections, we use an alternative integral quadratic cost function that can be formulated as follows:

$$J = \begin{cases} \int_{T_0}^{T_f} \alpha_1 \left(w_{s,A}(t) - w_{s,A,target}\right)^2 + \beta_1 (L(t) - L_{\max})^2 dt \quad if \ |w_s(t) - w_{s,A,target}| > 0.09 \\ \int_{T_0}^{T_f} \alpha_2 \left(w_{s,A}(t) - w_{s,A,target}\right)^2 + \beta_2 \left(L(t) - L_{\text{target}}\right)^2 dt \quad if \ |\, w_s(t) - w_{s,A,target}| \leq 0.09 \end{cases} \tag{3.4}$$

Formulation of cost in this form takes care of the sign of deviation and also leads to faster convergence to the target values because of the integral of squared deviation term. This can be attributed to the accumulation of the deviation cost over successive time steps due to the integral quadratic nature of the cost function. This is also evident from the results obtained in this work, discussed in section 5.2 later. The squared term is also important for avoiding singularities in the Jacobian computed as part of the optimization. A detailed discussion on such a choice of deviation based performance index is presented in [31]. As discussed in section 3.3.2 the parameters with subscript 's' have values scaled to lie in the domain [0,1] and the ratio of $\frac{\alpha_1}{\beta_1}$ is set to 4 when $\left|w_{s,A} - w_{s,A,target}\right| > 0.09$. Conversely, the ratio of parameters $\frac{\beta_2}{\alpha_2}$ is set to 4 when $\left|w_{s,A} - w_{s,A,target}\right| \leq 0.09$ in order to invert the weight on the components of the cost function. In effect this again fits into our generalized definition of cost with:

$$J_S = \int_{T_0}^{T_f} \alpha_i \left(w_{s,A}(t) - w_{s,A,target}\right)^2 dt \, , i \in \{1,2\} \tag{3.4a}$$

$$J_E = \int_{T_f}^{T_0} \beta_i (L(t) - L_{\max})^2 dt, i \in \{1,2\} \tag{3.4b}$$

where 3.4a represents the cost associated with the deviation of states from target values and 3.4b represents cost associated with the energy consumption. It is important to note that the level values in this performance index are not scaled. In this thesis the value of L_{max} is taken as 100%, and the value of L_{target} is taken as 64% which is as per the target steady-state value of level given in [1]. The reason for including the deviation of level from L_{max} when $\left|w_{s,A} - w_{s,A,target}\right| > 0.09$ and L_{target} when $\left|w_{s,A} - w_{s,A,target}\right| \leq 0.09$ in the cost function is same as the intuitive understanding discussed in section 3.3.2. This can also be explained in terms of rise time of the system.

In control theory low rise time for a step change is usually obtained for an under-damped second order system. The response obtained for level control as seen in Figure 5.5a later, is similar to step-response of a second-order under-damped system. Thus, choice of an integral quadratic PI for this level regulation problem can be viewed as augmenting the system with an integrator defined as the square of level deviation from the target values. This can lead to behavior of the system as an under-damped second order system with low rise-time.

Another important advantage of formulating an integral quadratic cost function for our solution approach is to transform the optimal control problem of the evaporation process into the framework of the Linear Quadratic Regulator [14] problem which is very widely used for optimal model predictive control of non-linear dynamical systems. In order to solve an optimization problem with state constraints a penalty function [31] can be added to the above integral quadratic cost function. The formulation of this penalty function is discussed later in section 5.1.

4. REFORMULATING THE MODEL

4.1 Mathematical modeling of a chemical process

Mathematical modeling of the transient operation of a process generally needs rigorous development based on physical principles and experimental parameter estimation. Conventional mathematical modeling of a chemical process is based upon the set of equations derived from the mass and energy balance of the process, the algebraic relationships between various process variables and the equations defining the physical constraints of respective state variables and inputs.

There are various forms of mathematical techniques for modeling MIMO (Multiple-Input-Multiple-Output) systems including ordinary differential equations, differential algebraic equations [2], relative gain arrays [17], etc. Hence, a particular chemical process can be modeled in multiple ways. Each method of modeling carries with it a set of assumptions made to simplify the analysis of the process. As discussed in [19], generally, the validity of a mathematical model is not given on a "true" or "false" basis, but rather on how accurately the equations represent the physical process and how the equations can be used to analyze the real behavior of the process. The mathematical model of the process thus developed can be further used for simulating the process and solving an optimization problem based on a specific performance metric (as discussed in Chapter 3) to compute the optimal control inputs. In this chapter, we reconsider the mathematical model set-forth in Chapter 2 and reformulate the set of differential algebraic equations as an interconnected dynamical system based on the methodology in the book [18]. In the following section 4.2 we provide a brief description of the interconnected hybrid dynamical system for modeling the evaporation system followed by a detailed reformulation of the evaporator model in the sections 4.3 and 4.4. The final two sections of the chapter discuss the details of the algorithm used in this work.

4.2 Introducing the interconnected hybrid dynamical system

Based on the formulation described in the work by RT Meyer and RA Decarlo [15],[18], the general form of an interconnected hybrid dynamical system for a process with two distinct modes can be give as:

$$\dot{x} = f_1(x, a_1) \tag{4.1}$$

$$b_1 = g_1(x, a_1) \tag{4.2}$$

$$a_1 = L_{11}^1 b_1 + L_{12}^1 u_1 \tag{4.3}$$

for mode 1, where x denotes vector of the state variables of the system having composite subsystem dynamics given by the function $f_1(x, a_1)$ for mode 1, b_1 is a vector of subsystem output variables in mode 1 and a_1 is a vector of input variables in mode 1. $g_1(x, a_1)$ is a non-linear function defining the algebraic relations of variables in b_1 to the states x and inputs a_1. The matrices L_{11}^1 and L_{12}^1 define the linear interconnection equations (for example, conservation of energy, power, Kirchhoff's current and voltage laws, conservation of mass, etc.) for a_1 to b_1 and overall system input u_1 for mode 1.

Similarly for mode 2 we have,

$$\dot{x} = f_2(x, a_2) \tag{4.4}$$

$$b_2 = g_2(x, a_2) \tag{4.5}$$

$$a_2 = L_{11}^2 b_2 + L_{12}^2 u_2 \tag{4.6}$$

where, x denotes the state variables of the system having composite subsystem dynamics given by the function $f_2(x, a_2)$ for mode 2, b_2 is a vector of subsystem output variables in mode 2 and a_2 is a vector of input variables in mode 2. On similar lines $g_2(x, a_2)$ is a non-linear function defining the algebraic relations of variables in b_2 to the states x and inputs a_2. The matrices L_{11}^2 and L_{12}^2 define the linear interconnection equations for a_2 to b_2 and overall system input u_2 for mode 2.

In order to simulate the system with the dynamics expressed in the above form using an iterative algorithm, for each time-step k, the equation (4.1) is used to propagate the states values and obtain x at time-step k given by $x\,(t_k)$ for mode 1 or likewise for mode 2 using equation (4.4) for the current time step using a predictor-corrector based algorithm [18], [35]. The predictor propagates state values in the respective mode using an explicit Euler formula as explained in [18]:

$$x\,(t_k) = x\,(t_{k-1}) + (t_k - t_{k-1})f_i\big(x\,(t_{k-1}), a_i(t_{k-1})\big) \tag{4.7}$$

where, t_k is the time at the k^{th} step of the iterative algorithm, $i = \{1 \text{ or } 2\}$ corresponding to the mode at time t_k. It is important to observe that the value of states for the k^{th} time step is calculated using the value of variables at the end of the previous time-step, precisely $x\,(t_{k-1})$ and $a_i(t_{k-1})$, in the predictor step. Also, when the mode for each time step is known, only the dynamics (f_1 or f_2) corresponding to the respective mode will be used for propagating the state values for this time-step. Note that the predictor step calculated using (4.7) is an approximation and hence a corrector is needed to correct the predicted values for a more accurate simulation. In our implementation we use an implicit collocation corrector [35] to correct the value of the predicted states $x(t_k)$ given by:

$$x\,(t_k) \;=\; x\,(t_{k-1}) + (t_k - t_{k-1})f_i\left(\frac{x\,(t_{k-1})+x\,(t_k)}{2}, a_{mp}\right) \tag{4.8}$$

where a_{mp} corresponds to the value of a_i variables computed corresponding to the value of $x = \frac{x\,(t_{k-1})+x\,(t_k)}{2}$. We do this by computing the value of a_{mp} as explained below.

Rearranging the system of equations given by equation (4.2), (4.3), (4.5) and (4.6) for computing a_{mp} and b_{mp}, we have the following equations for each mode:

$$b_{mp} - g_i\left(\frac{x\,(t_{k-1})+x\,(t_k)}{2}, a_{mp}\right) = 0 \tag{4.9a}$$

$$a_{mp} - L_{11}^{i} b_{mp} = L_{12}^{i} u_i(t_k) \tag{4.9b}$$

where $i = \{1 \text{or} 2\}$ corresponding to the mode for time-step k. We also know the input for this mode $u_i(t_k)$ that is assumed constant over each time-step as described later in section 4.5. The system of equations given by (4.9a) and (4.9b) is solved simultaneously for obtaining the values of the variables b_{mp} and a_{mp} corresponding to $x = \frac{x\,(t_{k-1})+x\,(t_k)}{2}$. This is done using a Newton Raphson algorithm and using Householder's formula as described in chapter-IV in [18].

The computed a_{mp} value is then used for correcting $x(t_k)$ using the corrector equation defined in equation (4.8). We then compute the corrected values of $b_i(t_k)$, $a_i(t_k)$ by simultaneously solving the following equations (4.10a) and (4.10b) which are similar to equations (4.9a) and (4.9b), using the same method as described above.

$$b_i(t_k) - g_i(x(t_k), a_i(t_k)) = 0 \tag{4.10a}$$

$$a_i(t_k) - L_{11}^i b_i(t_k) = L_{12}^i u_i(t_k) \tag{4.10b}$$

In the case of state dependent mode transitions, determination of the mode for $(k+1)^{th}$ time step can be made at this point based on the $x(t_k)$, $b_i(t_k)$ and $a_i(t_k)$ computed thus far. Likewise the system can be simulated for a given time span $[t_0, t_f]$ using the iterative process described above. The algorithm for our solution approach in this work is essentially formulated based on the above method as explained in the final section of this chapter.

In the following sections we re-formulate the evaporation system discussed in Chapter 2 in the general form of an interconnected hybrid dynamical system described above. Table 4.1 on the following page shows the variable mapping for reformulation of each mode. In section 4.3 we first reformulate the non-evaporating mode followed by reformulation of the evaporating mode in section 4.4. Section 4.5 presents a logical flow of the algorithm developed and lastly, section 4.6 talks in detail about the algorithm implemented.

Table 4.1 Variable mapping for evaporator simulation as interconnected hybrid dynamical system

	$\boldsymbol{xne}$	**Symbol**	**Variable in code**	$\boldsymbol{xe}$	
	$\boldsymbol{x_1}$	$\boldsymbol{m_A}$	**V.mA**	$\boldsymbol{x_1}$	
	$\boldsymbol{x_2}$	$\boldsymbol{m_B}$	**V.mB**	$\boldsymbol{x_2}$	
	$\boldsymbol{x_3}$	$\boldsymbol{m_C}$	**V.mC**	$\boldsymbol{x_3}$	
	$\boldsymbol{x_4}$	$\boldsymbol{U}$	**V.U**	$\boldsymbol{x_4}$	
$\boldsymbol{a_{ne}}$	$\boldsymbol{b_{ne}}$	**Symbol**	**Variable in code**	$\boldsymbol{a_e}$	$\boldsymbol{b_e}$
ane_1	$-$	$\boldsymbol{\dot{m}_{i,l}}$	**V.mil**	ae_1	$-$
ane_2	$-$	$\boldsymbol{\dot{m}_{o,l}}$	**V.$\boldsymbol{m}$ol**	ae_2	$-$
$-$	$-$	$\boldsymbol{\dot{m}_{o,v}}$	**V.mov**	ae_3	be_{13}
ane_3	bne_3	$\boldsymbol{\dot{m}_{HE}}$	**V.mhe**	ae_4	be_{14}
ane_4	bne_4	$\boldsymbol{Q}$	**V.Q**	ae_{15}	be_4
ane_{12}	bne_2	$\boldsymbol{P_{HE}}$	**V.Phe**	ae_{17}	be_{15}
ane_{13}	$-$	$\boldsymbol{T_{HE}}$	**V.The**	ae_{16}	$-$
ane_5	bne_5	$\boldsymbol{w_A}$	**V.wA**	ae_5	be_5
ane_6	bne_6	$\boldsymbol{w_B}$	**V.wB**	ae_6	be_6
ane_7	bne_7	$\boldsymbol{w_C}$	**V.wC**	ae_7	be_7
ane_8	bne_8	$\boldsymbol{T}$	**V.T**	ae_8	be_8
ane_{10}	bne_9	$\boldsymbol{P}$	**V.P**	ae_{18}	be_{12}
$-$	bne_{10}	$\boldsymbol{L}$	**V.L**	$-$	be_{11}
$-$	$-$	$\boldsymbol{\xi_B}$	**V.EB**	ae_9	be_9
$-$	$-$	$\boldsymbol{\xi_C}$	**V.EC**	ae_{10}	be_{10}
$-$	$-$	$\boldsymbol{\rho_v}$	**V.rhov**	ae_{12}	be_2
$-$	$-$	$\boldsymbol{V_v}$	**V.Vv**	ae_{13}	be_3
ane_{11}	bne_1	$\boldsymbol{m_l}$	**V.ml**	ae_{11}	be_1
$\boldsymbol{a_{ne}}$	$\boldsymbol{u_{ne}}$	**Symbol**	**Variable**	$\boldsymbol{a_e}$	$\boldsymbol{u_e}$
$-$	une_1	$\boldsymbol{V_1}$	**V.V1**	$-$	ue_1
$-$	une_2	$\boldsymbol{V_2}$	**V.V2**	$-$	ue_2
$-$	$-$	$\boldsymbol{V_{v1}}$	**V.Vv1**	ae_{19}	ue_3
ane_9	une_3	$\boldsymbol{V_{v2}}$	**V.Vv2**	ae_{14}	ue_4

4.3 Non-evaporating mode

In this section we consider the dynamics of the non-evaporating stage set forth in the Chapter 2. We reformulate the dynamics equations according to the variable mapping in Table 4.1. First we start with reformulating the state dynamic equations for the mass and energy balance for non-evaporating mode given as:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m_{o,l}} w_A \quad \text{(from 2.1)}$$

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m_{o,l}} w_B \quad \text{(from 2.2)}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m_{o,l}} w_C \quad \text{(from 2.3)}$$

$$\frac{dU}{dt} = \dot{m}_{i,l} c_{p,i,l}(w) T_i - \dot{m}_{o,l} c_{p,l}(w) T + \dot{Q} \quad \text{(from 2.4)}$$

These equations can be reformulated as:

$$\frac{dx_1}{dt} = ane_1 w_{A,i} - ane_2 ane_5 \quad (4.11)$$

$$\frac{dx_2}{dt} = ane_1 w_{B,i} - ane_2 ane_6 \quad (4.12)$$

$$\frac{dx_3}{dt} = ane_1 w_{C,i} - ane_2 ane_7 \quad (4.13)$$

$$\frac{dx_4}{dt} = ane_1 c_{p,i,l}(w) T_i - ane_2 c_{p,l}(w) ane_8 + ane_4 \quad (4.14)$$

here, as described in Appendix B the relationships for $c_{p,l}(w)$ and $c_{p,i,l}(w)$ are given by:

$$c_{p,i,l}(w) = 3.05 w_{A,i} + 4.315 w_{B,i} + 3.5 w_{C,i} \quad (4.14a)$$

$$c_{p,l}(w) = 3.05 ane_5 + 4.315 ane_6 + 3.5 ane_7 \quad (4.14b)$$

The relation equations for the process variables are given by the linear and nonlinear equations defined in Chapter 2. We pick specific variables as output variables '*b*' as described below and represent the corresponding relationship by recalling the specific equations from Chapter 2. The '*b*' variables for the non-evaporating mode are given as:

$$bne_1 : m_l = m_A + m_B + m_C \quad \text{(from 2.6)}$$

$$bne_2 : P_{HE} = \frac{T_{HE}}{T_{is}} P_i \quad \text{(from 2.23)}$$

$$bne_3 : \dot{m}_{HE} = -0.069 \sqrt{\rho_{v,i}} \frac{(P_{D,i}^3 - 36 P_{D,i}^2 - 69 P_{D,i})}{P_{D,i} + 0.08} V_{V2} \quad \text{(from 2.22)}$$

$$bne_4 : \dot{Q} = \left[c_{p,v,i} T_{is} - c_{p,l,i} T_{HE} + h_{V,i}\right] \dot{m}_{HE} \quad \text{(from 2.21)}$$

$$bne_5: w_A = m_A/m_l \quad \text{(from 2.7a)}$$

$$bne_6: w_B = m_B/m_l \quad \text{(from 2.7b)}$$

$$bne_7: w_C = m_C/m_l \quad \text{(from 2.7c)}$$

$$bne_8: T = U/(m_l c_{p,l}(w)) \quad \text{(from 2.9)}$$

$$bne_9: P = \left[\frac{w_B}{M_B} P_B^0(T) + \frac{w_C}{M_C} P_C^0(T)\right] / \left[\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right] \quad \text{(from 2.24)}$$

$$bne_{10}: L = \frac{\frac{m_l}{\rho_l(w)} - V_B}{v_L} \quad \text{(from 2.25)}$$

The reformulation of the above mentioned 'b' equations using the corresponding mapping of x, a and b from Table 4.1 is given by the following set of equations:

$$bne_1 = x_1 + x_2 + x_3 = g_1(x, ane) \quad (4.15)$$

$$bne_2 = \frac{P_i}{T_{is}} ane_{13} = g_2(x, ane) \quad (4.16)$$

$$bne_3 = -0.069 \sqrt{\rho_{v,i}} \frac{\left((P_i - ane_{12})^3 - 36(P_i - ane_{12})^2 - 69(P_i - ane_{12})\right)}{(P_i - ane_{12}) + 0.08} ane_9 = g_3(x, ane)$$

(4.17)

$$bne_4 = \left[c_{p,v,i} T_{is} - c_{p,l,i} ane_{13} + h_{V,i}\right] ane_3 = g_4(x, ane) \quad (4.18)$$

$$bne_5 = x_1/(ane_{11} + \varepsilon) = g_5(x, ane) \quad (4.19)$$

$$bne_6 = x_2/(ane_{11} + \varepsilon) = g_6(x, ane) \quad (4.18)$$

$$bne_7 = x_3/(ane_{11} + \varepsilon) = g_7(x, ane) \quad (4.21)$$

$$bne_8 = x_4/(ane_{11} c_{p,l}(w) + \varepsilon) = g_8(x, ane) \quad (4.22)$$

$$bne_9 = \left[\frac{ane_6}{M_B} P_B^0(ane_8) + \frac{ane_7}{M_C} P_C^0(ane_8)\right] / \left[\frac{ane_5}{M_A} + \frac{ane_6}{M_B} + \frac{ane_7}{M_C}\right] = g_9(x, ane) \quad (4.23)$$

$$bne_{10} = \frac{\frac{ane_{11}}{(1046.085\ ane_5 + 930.42\ ane_6 + 658.9\ ane_7)} - V_B}{v_L} = g_{10}(x, ane) \quad (4.24)$$

where according to the relationships given in Appendix B,

$$P_B^0(ane_8) = (0.0000050 ane_8^3 - 0.0048 ane_8^2 + 1.54 ane_8 - 164.9) \quad (4.24a)$$

$$P_C^0(ane_8) = (0.00001038 ane_8^3 - 0.0098 ane_8^2 + 3.14 ane_8 - 339.75) \quad (4.24b)$$

Note that in the above equations ε is a regularization constant added and takes a value of 0.001. This is done with a purpose of avoiding a divide by zero error and obtaining a stable solution. The linear interconnection equations ('a' equations) for the non-evaporating mode can be listed as follows:

$$ane_1 = \dot{m}_{i,l} = 10une_1 \tag{4.25}$$

$$ane_2 = \dot{m}_{o,l} = 10une_2 \tag{4.26}$$

$$ane_3 = \dot{m}_{HE} = bne_9 \tag{4.27}$$

$$ane_4 = Q = bne_4 \tag{4.28}$$

$$ane_5 = w_A = bne_5 \tag{4.29}$$

$$ane_6 = w_B = bne_6 \tag{4.30}$$

$$ane_7 = w_C = bne_7 \tag{4.31}$$

$$ane_8 = T = bne_8 \tag{4.32}$$

$$ane_9 = V_{v2} = une_3 \tag{4.33}$$

$$ane_{10} = P = bne_9 \tag{4.34}$$

$$ane_{11} = m_l = bne_1 \tag{4.35}$$

$$ane_{12} = P_{HE} = bne_2 \tag{4.36}$$

$$ane_{13} = T_{HE} = \frac{bne_4}{K.A} + bne_8 \qquad \text{(from 2.20)}$$

Thus we have an interconnected dynamical model of the non-evaporating mode expressed in the standard form:

$$\dot{x} = f_{ne}(x, ane) \tag{4.37}$$

$$bne = g_{ne}(x, ane) \tag{4.38}$$

$$ane = Lne_{11}bne + Lne_{12}une \tag{4.39}$$

where,
$$Lne_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{1}{K.A}\right) & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.40}$$

$$Lne_{12} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.41}$$

4.4 Evaporating mode

In this section we reformulate the equations governing the evaporating mode as an interconnected dynamical system. The equations for this mode are set-forth in chapter 2. It is to be noted that the vapor phase is modeled with the relationship for the density of vapor (ρ_v) given by an additional equation given in the report by C. Sonntag et. al. [2]:

$$\rho_v = \frac{w_B P_B^0(T) + w_C P_C^0(T)}{RT\left(\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right)} \tag{4.42}$$

We now recall the state dynamic equations for the mass and energy balance for the evaporating mode given by:

$$\frac{dm_A}{dt} = \dot{m}_{i,l} w_{A,i} - \dot{m}_{o,l} w_A \qquad \text{(from 2.10)}$$

$$\frac{dm_B}{dt} = \dot{m}_{i,l} w_{B,i} - \dot{m}_{o,l} w_B - \dot{m}_{o,v}\, \xi_B \qquad \text{(from 2.11)}$$

$$\frac{dm_C}{dt} = \dot{m}_{i,l} w_{C,i} - \dot{m}_{o,l} w_C - \dot{m}_{o,v}\, \xi_C \qquad \text{(from 2.12)}$$

$$\frac{dU}{dt} = \dot{m}_{i,l} c_{p,i,l}(w) T_i - \dot{m}_{o,l} c_{p,l}(w) T - \dot{m}_{o,v} f_v(T,\xi) + \dot{Q} \quad \text{(from 2.13)}$$

According to the mapping in Table 4.1 these equations can be reformulated as:

$$\frac{dx_1}{dt} = ae_1 w_{A,i} - ae_2 ae_5 \quad (4.43)$$

$$\frac{dx_2}{dt} = ae_1 w_{B,i} - ae_2 ae_6 - ae_3 ae_9 \quad (4.44)$$

$$\frac{dx_3}{dt} = ae_1 w_{C,i} - ae_2 a_7 - ae_3 ae_{10} \quad (4.45)$$

$$\frac{dx_4}{dt} = ae_1 c_{p,i,l}(w) T_i - ae_2 c_{p,l}(w) ae_8 - ae_3 f_v(\xi,T) + ae_4 \quad (4.46)$$

here, as described in Appendix B the relationships for $c_{p,l}(w)$, $c_{p,i,l}(w)$ and $f_v(\xi,T)$ are given by:

$$c_{p,i,l}(w) = 3.05 w_{A,i} + 4.315 w_{B,i} + 3.5 w_{C,i} \quad (4.46a)$$

$$c_{p,l}(w) = 3.05 ae_5 + 4.315 ae_6 + 3.5 ae_7 \quad (4.46b)$$

$$f_v(\xi,T) = (0.0019325 ae_9 + 0.0016115 * ae_{10}) ae_8 + (2375.8752 ae_9 + 1184.8742 ae_{10}) \quad (4.46c)$$

The relation equations for the process variables are given by the linear and nonlinear equations defined in Chapter 2. Here we list the output variables ‘b’ as described below and represent the corresponding relationship by recalling the specific equations from Chapter 2. The ‘b’ variables for the evaporating mode are given as:

$$be_1: m_l = m_A + m_B + m_C - \rho_v V_v \quad \text{(from 2.16a,b,c)}$$

$$be_2: \rho_v = (w_B P_B^0(T) + w_C P_C^0(T)) / \left(RT\left(\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right)\right) \quad \text{(from 4.42)}$$

$$be_3: V_v = V - \frac{m_l}{\rho_l(w)} \quad \text{(from 2.17)}$$

$$be_4: \dot{Q} = \left[c_{p,v,i} T_{is} - c_{p,l,i} T_{HE} + h_{V,i}\right] \dot{m}_{HE} \quad \text{(from 2.21)}$$

$$be_5: w_A = m_A / m_l \quad \text{(from 2.16a)}$$

$$be_6: w_B = (m_B - \xi_B \rho_v V_v) / m_l \quad \text{(from 2.16b)}$$

$$be_7: w_C = (m_C - \xi_C \rho_v V_v) / m_l \quad \text{(from 2.16c)}$$

$$be_8: T = (U - h_v(\xi) \rho_v V_v) / (m_l c_{p,l}(w) + c_{p,v}(\xi) \rho_v V_v) \quad \text{(from 2.19)}$$

$$be_9: \xi_B = \frac{w_B P_B^0(T)}{w_B P_B^0(T) + w_C P_C^0(T)} \quad \text{(from 2.18a)}$$

$$be_{10}: \xi_C = \frac{w_C P_C^0(T)}{w_B P_B^0(T) + w_C P_C^0(T)} \quad \text{(from 2.18b)}$$

$$be_{11}: L = \frac{\frac{m_l}{\rho_l(w)} - V_B}{v_L} \quad \text{(from 2.25)}$$

$$be_{12}: P = \left[\frac{w_B}{M_B} P_B^0(T) + \frac{w_C}{M_C} P_C^0(T)\right] / \left[\frac{w_A}{M_A} + \frac{w_B}{M_B} + \frac{w_C}{M_C}\right] \quad \text{(from 2.24)}$$

$$be_{13}: \dot{m}_{o,v} = \left[-0.069\ \sqrt{\rho_v(w,T)}\frac{(P_D^3 - 36P_D^2 - 69P_D)}{P_D + 0.08}\right] V_{V1} \quad \text{(from 2.14)}$$

$$be_{14}: \dot{m}_{HE} = \left[-0.069\ \sqrt{\rho_{v,i}}\frac{(P_{D,i}^3 - 36P_{D,i}^2 - 69P_{D,i})}{P_{D,i} + 0.08}\right] V_{V2} \quad \text{(from 2.22)}$$

$$be_{15}: P_{HE} = \frac{P_i}{T_i} T_{HE} \quad \text{(from 2.23)}$$

The reformulation of these equations can be given by the following set of equations:

$$be_1 = x_1 + x_2 + x_3 - ae_{12}\ ae_{13} = g_1(x, ae) \quad (4.47)$$

$$be_2\ = (ae_6 P_B^0(ae_8) + ae_7 P_C^0(ae_8))/(Rae_8\left(\frac{ae_5}{M_A} + \frac{ae_6}{M_B} + \frac{ae_7}{M_C}\right)) = g_2(x, ae) \quad (4.48)$$

$$be_3 = V - \frac{ae_{11}}{(1046.085\ ae_5 + 930.42\ ae_6 + 658.9\ ae_7)} = g_3(x, ae) \quad (4.49)$$

$$be_4 = \left[c_{p,v,i} T_{is} - c_{p,l,i}\ ae_{16} + h_{V,i}\right] ae_4 = g_4(x, ae) \quad (4.50)$$

$$be_5 = \ x_1/(ae_{11} +\ \varepsilon) = g_5(x, ae) \quad (4.51)$$

$$be_6 = (x_2 - ae_9\ ae_{12}\ ae_{13}\ )/(ae_{11} +\ \varepsilon) = g_6(x, ae) \quad (4.52)$$

$$be_7\ =\ (x_3 - ae_{10}\ ae_{12}\ ae_{13}\ )/(ae_{11} +\ \varepsilon) = g_7(x, ae) \quad (4.53)$$

$$be_8 = (x_4 - h_v(\xi)\ ae_{12} ae_{13}\ )/(ae_{11}\ c_{p,l}(w) + c_{p,v}(\xi) ae_{12} ae_{13} +\ \varepsilon) = g_8(x, ae) \quad (4.54)$$

$$be_9 = \frac{ae_6 P_B^0(ae_8)}{ae_6 P_B^0(ae_8) + ae_7 P_C^0(ae_8)} = g_9(x, ae) \quad (4.55)$$

$$be_{10} = \frac{ae_7 P_C^0(ae_8)}{ae_6 P_B^0(ae_8) + ae_7 P_C^0(ae_8)} = g_{10}(x, ae) \quad (4.56)$$

$$be_{11}\ = \frac{\frac{ae_{11}}{(1046.085\ ae_5 + 930.42\ ae_6 + 658.9\ ae_7)} - V_B}{v_L} = g_{11}(x, ae) \quad (4.57)$$

$$be_{12} =\ \left[\frac{ae_6}{M_B} P_B^0(ae_8) + \frac{ae_7}{M_C} P_C^0(ae_8)\right] / \left[\frac{ae_5}{M_A} + \frac{ae_6}{M_B} + \frac{ae_7}{M_C}\right] = g_{12}(x, ae) \quad (4.58)$$

$$be_{13} = -0.069\ \sqrt{ae_{12}}\frac{((ae_{18} - P_A)^3 - 36(ae_{18} - P_A)^2 - 69(ae_{18} - P_A)\ )}{(ae_{18} - P_A)\ + 0.08}\ ae_{19} = g_{13}(x, ae) \quad (4.59)$$

$$be_{14} = -0.069\ \sqrt{\rho_{v,i}}\frac{\left((P_i - ae_{17})^3 - 36(P_i - ae_{17})^2 - 69(P_i - ae_{17})\right)}{(P_i - ae_{17}) + 0.08} ae_{14}\ = g_{14}(x, ae) \quad (4.60)$$

$$be_{15} = \frac{P_{is}}{T_{is}}\ ae_{16}\ = g_{15}(x, ae) \quad (4.61)$$

where, according to the relationships given in Appendix B,

$$P_B^0(ae_8) = (0.0000050ae_8^3 - 0.0048ae_8^2 + 1.54ae_8 - 164.9) \quad (4.61a)$$

$$P_C^0(ae_8) = (0.00001038ae_8^3 - 0.0098ae_8^2 + 3.14ae_8 - 339.75) \quad (4.61b)$$

The interconnection equations for this mode can be listed as follows:

$$ae_1 = \dot{m}_{i,l} = 10ue_1 \quad \text{(from 2.1a)}$$

$$ae_2 = \dot{m}_{o,l} = 10ue_2 \quad \text{(from 2.1b)}$$

$$ae_3 = \dot{m}_{o,v} = be_{13} \quad (4.62)$$

$$ae_4 = \dot{m}_{HE} = 10ue_2 \quad (4.63)$$

$$ae_5 = w_A = be_5 \quad (4.64)$$

$$ae_6 = w_B = be_6 \quad (4.65)$$

$$ae_7 = w_C = be_7 \quad (4.66)$$

$$ae_8 = T = be_8 \quad (4.67)$$

$$ae_9 = \xi_B = be_9 \quad (4.68)$$

$$ae_{10} = \xi_C = be_{10} \quad (4.69)$$

$$ae_{11} = m_l = be_1 \quad (4.70)$$

$$ae_{12} = \rho_v = be_2 \quad (4.71)$$

$$ae_{13} = V_v = be_3 \quad (4.72)$$

$$ae_{14} = V_{v2} = be_2 \quad (4.73)$$

$$ae_{15} = \dot{Q} = be_4 \quad (4.74)$$

$$ae_{16} = T_{HE} = \frac{be_4}{K.A} + be_8 \quad \text{(from 2.20)}$$

$$ae_{17} = P_{HE} = be_{15} \quad (4.75)$$

$$ae_{18} = P = be_{12} \quad (4.76)$$

$$ae_{19} = V_{v1} = ue_3 \quad (4.77)$$

Thus we have an interconnected dynamical model of the evaporating mode expressed in the standard form:

$$\dot{x} = f_e(x, ae) \quad (4.78)$$

$$be = g_e(x, ae) \quad (4.79)$$

$$ae = Le_{11}be + Le_{12}ue \quad (4.80)$$

$$\text{where,}\quad Le_{11}=\begin{bmatrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1/k.A&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{bmatrix}\tag{4.81}$$

$$Le_{12}=\begin{bmatrix}10&0&0&0\\0&10&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{bmatrix}\tag{4.82}$$

4.5 Logical flow of the algorithm developed

The interconnected hybrid dynamical model developed in this chapter is solved using an NMPC optimal control algorithm developed in this research work. This algorithm uses a predictor-corrector based relaxation algorithm for propagating the state dynamics followed by a Newton Raphson algorithm implementation for simultaneously solving for the a's and b's as described in section 4.2. The algorithm used to solve the optimization problem is a new version of the existing algorithms in which the model constraints and cost function computation are performed outside of the sequential quadratic (optimization) program inside $fmincon$ from MATLAB. In this approach the model is solved using simulation as a set of constraints for the $fmincon$ optimization which computes the optimal sequence of inputs to minimize the objective function as described in chapter 3. Thus the developed solver additionally gives us the flexibility to vary the simulation step size. This overall flow of the algorithm is illustrated in Figure 4.1 below. In addition to the explanations given above and in the prior sections, we note that to simulate the measurement required for MPC in a "real" system we introduce a high fidelity simulation as a proxy. Thus the state values of the high fidelity computation initialize the optimization over the next MPC window.

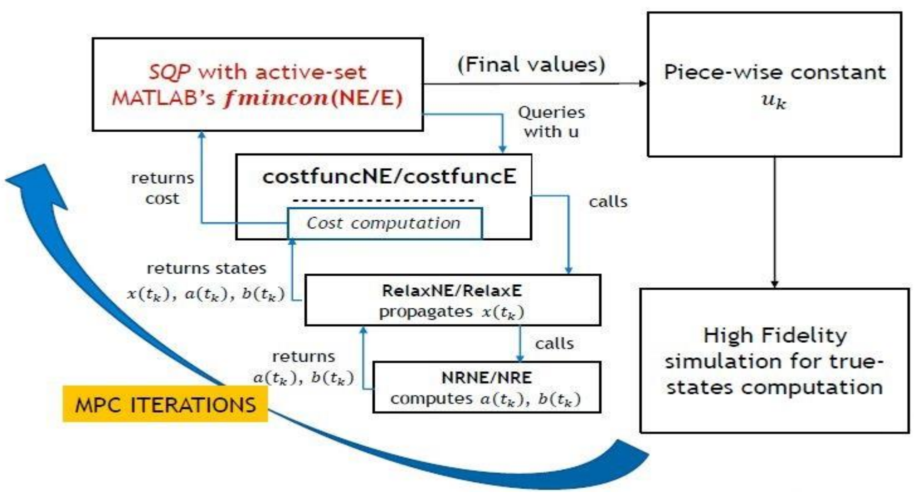


Figure 4.1 Block diagram of the algorithm implemented

As mentioned, our optimization approach is based on MPC. In model predictive control we solve for locally optimal controls over a prediction window. In our case the prediction window can either have a single or multiple partitions. The time width of each partition of a window is represented by h_{opt}. Thus if we have a 2 partition window, then the prediction window will have a time width of 2*h_{opt}; for 4 partition window the prediction window will have a time width of 4*h_{opt} and so on. Thus it follows that the number of optimizations we need to perform for a given final time $t_f$can be calculated as $N = t_f/h_{opt}$. For these optimizations the sampling time of simulations done in each partition of a window is given by the variable h_{sim}. Hence we have the relationship $h_{sim} = \frac{h_{opt}}{2}$ if we simulate for two time steps within each partition of the prediction window, $h_{sim} = \frac{h_{opt}}{100}$ if we simulate for 100 time steps within each partition of the prediction window. Next, we present an example to clarify this notion of timing. Consider the case when $t_f = 4$, $h_{opt} = 0.1$, $h_{sim} = 0.05$ and we consider a two partition window. In this case the solver will start solving for two optimal inputs, one from 0s to 0.1s and another from 0.1s to 0.2s in the first iteration by simulating the system starting from 0s to 0.2s in 4 steps $[0, 0.05, 0.1, 0.15]$. In the second iteration the solver will solve for two optimal inputs, one from 0.1s to 0.2s and another from 0.2s to 0.3s and so on till the two optimal inputs for the window from 3.9s to 4.1s is computed. Figure 4.2 depicts this clearly.

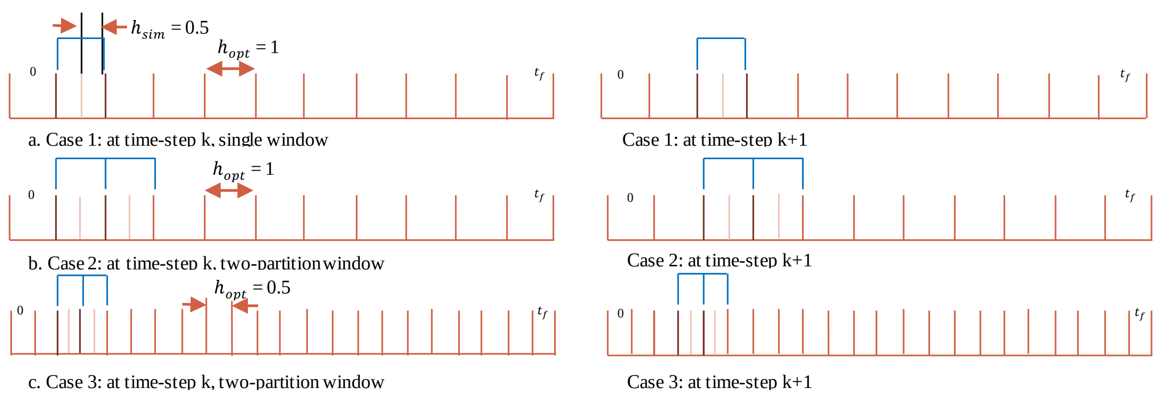


Figure 4.2 Time-step for a) single-window and b), c) two-partition-window NMPC

4.6 Discussion of the coding in the algorithm

In this section we present the logical flow of the code developed for this purpose by explaining the flow-diagram of the developed program. The logical flow of the algorithm is given in Figure 4.4. The code begins with a series of initialization for matrices for the non-evaporating mode and evaporating mode to perform this computation. This includes the matrices $Lne_{11}, Lne_{12}, Le_{11}$and Le_{12} as described in sections 4.3 and 4.4 and other matrices used for the Newton Raphson algorithm which remain unchanged throughout the program. Pre-allocation of matrices and use of structure to pass the variables to functions in MATLAB was found to be a fast approach to store data in the process of simulation and optimization. The structure named V holds value for all the system variables for each time step of the iterative process. A total of 26 variables are contained in the structure V and the variables are named as given in Table 4.1. In the predictor corrector approach the value of state at next time step is obtained by propagating state dynamics for one time step starting from an initial value.

The algorithm for hybrid optimal control of evaporation system is implemented in a file called mainEvapSolver.m and is depicted in Figure 4.4.

In step 1 we first initialize the time related information by setting values of t_0, t_f, h_{opt}, h_{sim}, Npart, Nhires where, Npart gives the number of partitions of the prediction window and Nhires gives number of partitions for the high fidelity simulation, which will be discussed later in this section.

In step 2 we initialize a structure of constant parameters for the system denoted as C. This contains a set of all constants used in this code like, molecular mass of liquid components, specific heat capacity of fresh steam, heat exchanger parameters etc. The list of parameters can be known by referring to the beginning of the code in Appendix-C.

In step 3 we initialize the set of matrices needed for computations in the non-evaporating mode. These matrices are stored in the structure named SNE that is passed to the functions for the non-evaporating mode.

In step 4 we do a similar initialization for matrices as in step 3 for the evaporating mode in the structure SE. SE is passed to the functions for the evaporating mode.

In step 5 we initialize the process variables to their starting values. The details of the initial values of all the process variables are given in Table 5.1. Note that in this step une_guess is the initial guess of input values for the non-evaporating mode given as a vector of [V.V1; V.V2; V.Vv2] and ue_guess is the initial guess of input values for the evaporating mode given as a vector of [V.V1; V.V2;V.Vv1;V.Vv2]. It is important to note at this point that the optimization of non-evaporating mode is over the set of the three valves V_1, V_2 and V_{v2} as described in section 1.2 and the optimization for the evaporating mode is over the set of 4 valves V_1, V_2, $V_{v1}$and V_{v2}. Precisely, the vapor out-flow valve is never operated in the non-evaporating mode due to absence of vapor phase in the evaporation column.

In step 6 we initialize the upper and lower bounds for the optimization using the $fmincon$ function. Later these are passed as the lb and ub variables in the input arguments to the $fmincon$ function. In this step we also initialize the initial mode of the system as non-evaporating mode by setting a variable mode to zero.

In step 7 we start iterating over each time step using k as the iteration variable running from 1 till N $(= t_f/h_{opt})$ as defined at the beginning of this section.

In step 8 we check if the mode for step k is zero. In this case the algorithm branches to execute a set of functions related to the non-evaporating mode; denoted as step 9.

In step 9, the $fmincon$ based solver starts from the value of guess for value of input, une_guess and calls the cost function corresponding to the non-evaporating mode, costfuncNE. This function returns the value of the cost function for this time-step computed using equation 5.5 using the value of the process variables obtained by calling the relaxation algorithm function RelaxNE for the non-evaporating mode. This is depicted in the block diagram of the algorithm in Figure 4.1. The relaxation function propagates the value of state variables using the state dynamics and calls the Newton Raphson algorithm function for the non-evaporating mode NRNE for simultaneously solving for the ane and bne as explained in section 4.2 and depicted in Figure 4.3. It is to be noted that the Relaxation function divides the time-step into two (Npart = 2) and thus makes two computations of states per time step as explained at the beginning of this section. The $fmincon$ function calls the function costfuncNE multiple times in this step to compute the optimal inputs $uv\_opt(k)$ for this time-step k. The solver computed values for the optimal inputs are then used to simulate the system over the k^{th} time-step by calling the RelaxNE function again. In this call the relaxation function computes the process variables by dividing the time step into 10 (Nhires) and hence giving a more accurate simulation of states and hence this is a high-fidelity simulation. The algorithm then proceeds to step 12 for updating the computed values in the structure V.

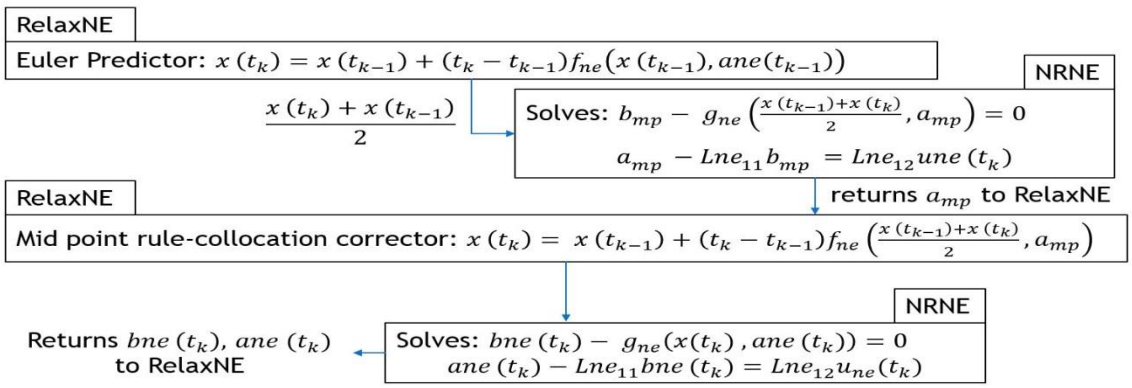


Figure 4.3 Execution of functions RelaxNE and NRNE

Step 10 will occur only if the mode for this time-step was not zero. Thus in this step the algorithm would branch to step 11 for executing the functions related to the evaporating mode. Consequently in step 11, similar to step 9, the $fmincon$ function starts from the value of guess for value of input, ue_guess and calls the cost function corresponding to the evaporating mode, costfuncE. This function returns the value of the cost function for this time step computed using equation 5.5 using the value of the process variables obtained by calling the relaxation algorithm function RelaxE for the evaporating mode. The relaxation function propagates the value of state variables using the state dynamics and calls the Newton Raphson algorithm function for the evaporating mode NRE for simultaneously solving for the ae and be as explained in section 4.2. It is to be noted again that the Relaxation function divides the time-step into two (Npart = 2) and thus makes two computations of states per time step as explained at the beginning of this section. The $fmincon$ function calls the function costfuncE multiple times in this step to compute the optimal inputs $uv\_opt(k)$ for this time-step k. The solver computed values for the optimal inputs are then used to simulate the system over the k^{th} time-step by calling the RelaxE function again. In this call the relaxation function computes the process variables by dividing the time step into 10 (Nhires) and hence giving a more accurate high-fidelity simulation of states. The algorithm then proceeds to step 12 for updating the computed values in the structure V.

In step 13 the decision for mode (NE:0, E:1) for the next-time step is made based on the value of pressure $V.P(k+1)$ at the end of the iteration and depending on the mode, the process variables are reset for the next time-step in step 14 or step 15 and the algorithm continues to step 7 for the next time step. Consequently in the next time-step of the iteration the value for the process variables are computed according to the updated mode value from the previous iteration starting from the values of process variables of the last time step stored in the structure V. In the end the structure V contains the optimal trajectories of all process variables and the inputs

.

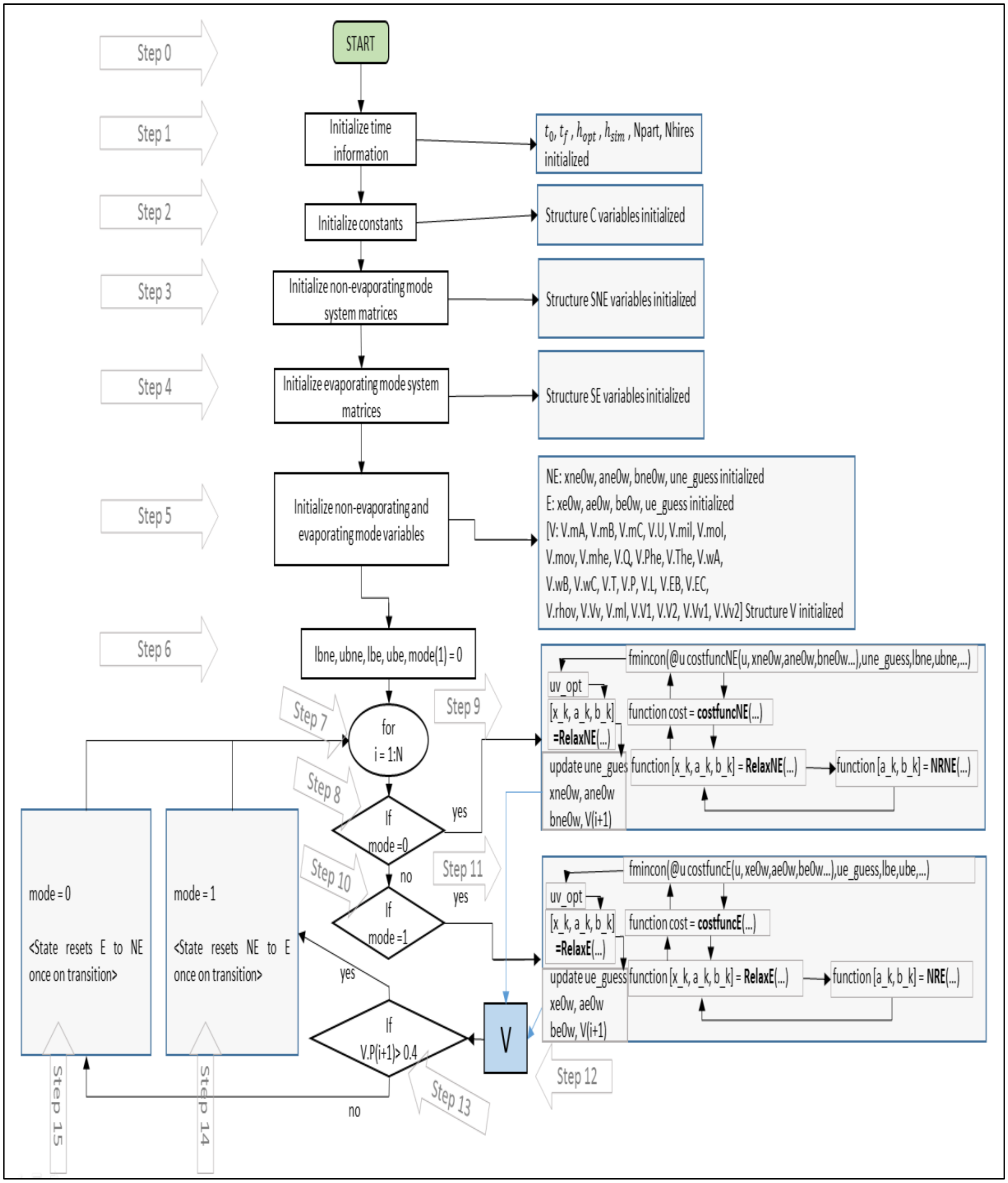


Figure 4.4 Algorithm [4] of mainEvapSolver.m

[4] This algorithm is collectively developed by Raymond DeCarlo, Rick Meyer, Scott Johnson and Rithesh Iyer and it is still under revision and improvement.

5. RESULTS AND DISCUSSIONS

In this section we first present the details of our implementation including the details of the performance index formulated, difference in implementation in our approach as compared to the approach used in [1] and finally the results of the optimal control approach describing the advantages gained in modeling the evaporation system as an interconnected dynamical system on the lines of the work done by RT Meyer and RA DeCarlo et.al in [15],[16],[36]. MATLAB 2013b licensed under ECN, Purdue University is used for implementing and compiling the results of this solution approach. First we begin with describing the formulation of the performance index for this optimization problem in the next section.

5.1 Implementation of the performance index

As discussed in section 3.3.3 the performance index for the optimization problem is formulated as an integral quadratic cost function. In case of the absolute deviation function as described in section 3.3.2, the Jacobian might take a singular value when the process variables cross the target values. Hence in order to avoid singularities in the Jacobian in the optimization and to lead to faster convergence, the integral quadratic performance index for our implementation is given as:

$$J = \begin{cases} \int_{T_0}^{T_f} \alpha_1\left(w_{s,A}(t) - w_{s,A,target}\right)^2 + \beta_1(L(t) - L_{\max})^2 dt \quad if\ |w_s(t) - w_{s,A,target}| > 0.09 \\ \int_{T_0}^{T_f} \alpha_2\left(w_{s,A}(t) - w_{s,A,target}\right)^2 + \beta_2\left(L(t) - L_{\text{target}}\right)^2 dt \quad if\ |\, w_s(t) - w_{s,A,target}| \leq 0.09 \end{cases}$$

$(from\ 3.4)$

The parameters α_1 and β_2 in our implementation are 4000 and α_2 and β_1 1000 respectively. Thus the square of deviation of level is weighted more when the concentration of component A in product is close to the target concentration of 0.8 and vice-versa. To implement this condition we check the value of the difference in the target concentration of 0.8 and the concentration of component A ($V.wA$) and assign the weights based on whether $|w_s(t) - w_{s,A,target}| > 0.09$ or $| w_s(t) - w_{s,A,target}| \leq 0.09$, inside the cost functions costfuncNE and costfuncE for the non-evaporating mode and evaporating mode respectively. This formulation of the cost function is also found to give faster convergence possibly due to the cumulative effect of cost over the time due to the integral term.

In addition to the above mentioned performance index we add penalty functions [9],[31] to the cost function to implement state constraints. This is done in order to penalize the deviation of variables from the upper and lower bounds of the process variables as described below. What is not done in this investigation is penalizing rate constraints on the control input changes. This is a topic for future work.

Nevertheless, in order for the process to operate under safe operating limits the penalty functions are implemented as squared penalty functions [9] for the following variables, the level inside the evaporator ($V.L$), pressure of vapor inside the evaporator ($V.P$), temperature inside the evaporator ($V.T$) and temperature of the heat exchanger surface ($V.T_{HE}$). Following are the values of the state bounds given in [2]:

$$0\% \leq L \leq 100\% \tag{5.1}$$

$$330K \leq T \leq 470K \tag{5.2}$$

$$0bar \leq P \leq 5bar \tag{5.3}$$

$$330K \leq T_{HE} \leq 470K \tag{5.4}$$

Thus the overall cost function is formulated as:

$$\begin{aligned} Total\ cost = J + W.sq[(\Delta L_{ub})^2 + (\Delta P_{ub})^2 + (\Delta T_{ub})^2 + \left(\Delta T_{HE_{ub}}\right)^2] \\ +W.sq[(\Delta L_{lb})^2 + (\Delta P_{lb})^2 + (\Delta T_{lb})^2 + \left(\Delta T_{HE_{lb}}\right)^2] \end{aligned} \tag{5.5}$$

where, subscript ub stands for violation in upper bound and subscript lb stands for violation in lower bound, ΔL gives violation of level of liquid in the evaporator, ΔP gives violation of pressure of vapor inside the evaporator, ΔT gives violation of temperature inside the evaporator, ΔT_{HE} gives violation of temperature of the surface of the heat exchanger. The term $W.sq$ stands for weights of squared deviation of states. For implementation, $W.sq$ is considered to be 1000. It can be noted from results later in this chapter, that the states remain bound within the limits for the whole duration of time for which the problem is solved. Hence these penalty constraints are never activated.

5.2 Test cases and results of the implementation

The code implementing the algorithm in Figure 4.4 is given in Appendix-C. The code was tested with five cases: single-window NMPC approach with $h_{opt} = 1, h_{sim} = 0.5$, two-partition-window NMPC approach with $h_{opt} = 1$, $h_{sim} = 0.5$, two-partition-window NMPC approach with $h_{opt} = 0.5$, $h_{sim} = 0.25$, as explained in Figure 4.2. The additional two cases considered are single-window NMPC approach with $h_{opt} = 100$, $h_{sim} = 1$ and the case where we use parameters corresponding to superheated steam. In this section, we first present the results of our implementation for the first case and compare the results obtained using our solver with the results obtained in [1] using a NLP solver. Later, we discuss and contrast the results of the other cases.

Table 5.1 lists the initialized values of process variables and inputs similar to [1], used for running the computation. In effect only the initial values of inputs, level, temperature, heat transferred from heat exchanger, mass inflow from steam valve and concentrations of components of inflowing liquid need to be specified explicitly. Remaining variables are calculated from the equations in Chapter 2. The initial values explicitly specified are:

$$[V.wA, V.wB, V.L, V.T, V.Q, V.mhe, V.V1, V.V2, V.Vv1, V.Vv2] = [0.12, 0.85, 1, 327, 8000, 9, 1, 0, 0, 0.8]$$

Table 5.1 Variable Initialization

Variable	**Initialized value**	**Variable**	**Initialized value**
$V.mA(1)$	1129.9	$V.wC(1)$	0.03
$V.mB(1)$	800.35	$V.T(1)$	327
$V.mC(1)$	282.47	$V.P(1)$	0.2819
$V.U(1)$	$1.2743 * 10^7$	$V.L(1)$	1%
$V.mil(1)$	10	$V.EB(1)$	0.7743
$V.mol(1)$	0	$V.EC(1)$	0.2257
$V.mov(1)$	0	$V.rhov(1)$	0.2261
$V.mhe(1)$	9	$V.Vv(1)$	29.95
$V.Q(1)$	8000	$V.ml(1)$	9415.8
$V.Phe(1)$	3.49	$V.V1(1)$	1
$V.The(1)$	333K	$V.V2(1)$	0
$V.wA(1)$	0.12	$V.Vv1(1)$	1
$V.wB(1)$	0.85	$V.Vv2(1)$	0.8

5.2.1 Case 1: Single window NMPC results

In this section we discuss the results of the single-window NMPC approach using our solver described in section 4.5, and present a thorough comparison with the results obtained by C Sonntag et. al. in [1] using the NLP solver. First we present the sequence of controls computed by the solver over the time interval of [0,15000s] for the single-window case with h_{opt} = 1, h_{sim} = 0.5. Later in this section we will look into the behavior of other process variables in this case.

5.2.1.1 Inlet feed control valve $V1$

Figure 5.1a, depicts the plot for the computed valve operation sequence for the feed inflow valve $V1$. It suggests opening the valve at 100% in the beginning till the level reaches its maximum value of 100% and controlling the valve in the range of about 64% to 65% opening thereafter till the target concentration of component A in product is attained. Thereafter the valve can be operated in the range of 50%-53% to maintain a steady flow of product at the desired concentration as seen in the Figure 5.1a from about 11940s till 15000s. It can be observed that the sequence of control computed by Sontag et. al. [1] using their NLP solver for valve V_1 as shown in Figure 5.1b is very similar. It is found that at about 11663s the concentration of A in the product attains the value of target value minus 0.09 (0.71), which is the time after which the level is brought down to the steady state value of 64%. Hence the inlet feed valve is closed at that time. In Figure 5.1b this time corresponds to about 13000s. At 11940s the valve is opened again to about 45-50% opening after which it is maintained in the range of 50-53% opening. This opening of the valve corresponds to about 13200s in Figure 5.1b.

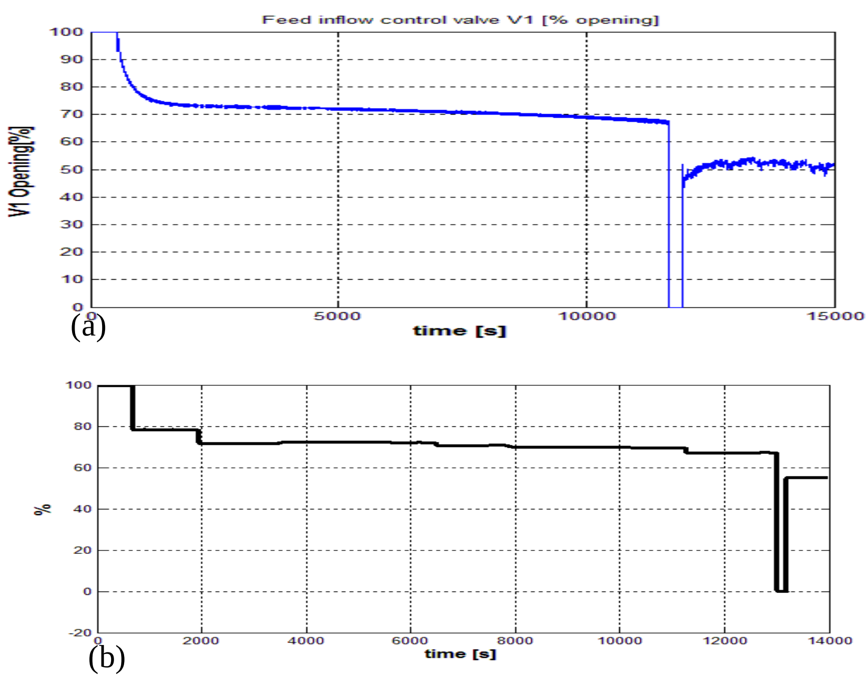


Figure 5.1 Inlet feed control valve operation a) our solver b) NLP solver in [1]

5.2.1.2 Product outflow control valve $V2$

Figure 5.2 depicts the control for the product outflow valve. The product outflow valve is kept at a closed position till the target concentration of component A in product is reached. The plot shows oscillations about a mean value close to 8% opening after the product target concentration is attained at about 11921s, indicated by the dense blue areas in the plot. Thus the product outflow valve can be opened in the range of [8%, 8.25%] to dispatch the product at around 11921s. The control is very noisy in this case and averages to a value of about 8.15% opening. Such a noisy control can be handled by passing the control signal through a low pass filter to remove the high frequency oscillations before the signal is given to operate the valves in the real-system or to the high-fidelity simulation in our case. Another way to avoid oscillations in computation is to penalize rate constraints on input controls as discussed in section 5.1.

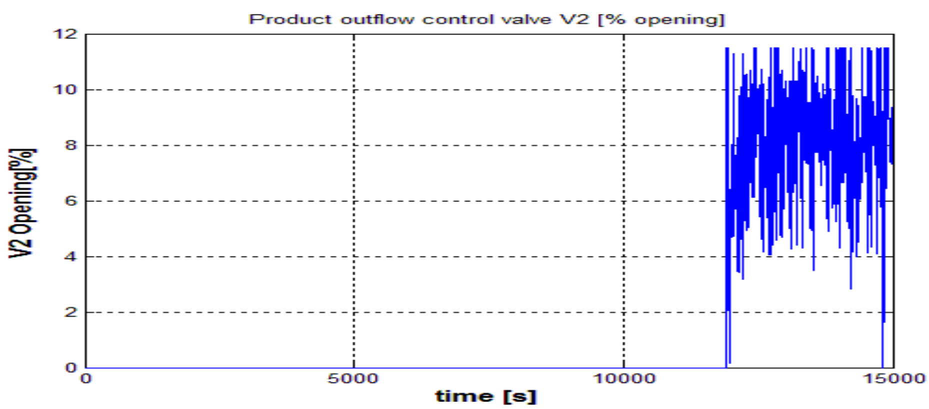


Figure 5.2 Product outflow control valve

5.2.1.3 Steam inflow valve Vv_2

Figure 5.3 suggests that the steam inflow valve into the heat exchanger must be closed before the target level is reached and then opened to 100% opening position till 11663s when the concentration in product A is near the target value. Since this valve is assumed to be operated in the continuous range of [80%, 100%] opening, the steady state opening of this valve is computed to be about 86%-90%. This drop in steam valve opening at steady-state operation can be attributed to reduced level of liquid in the column which would require relatively less amount of heat transferred to the evaporation column to maintain a constant rate of evaporation.

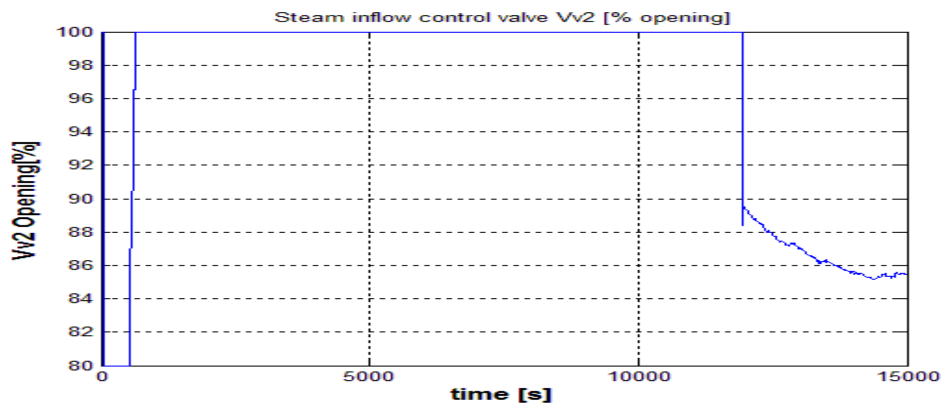
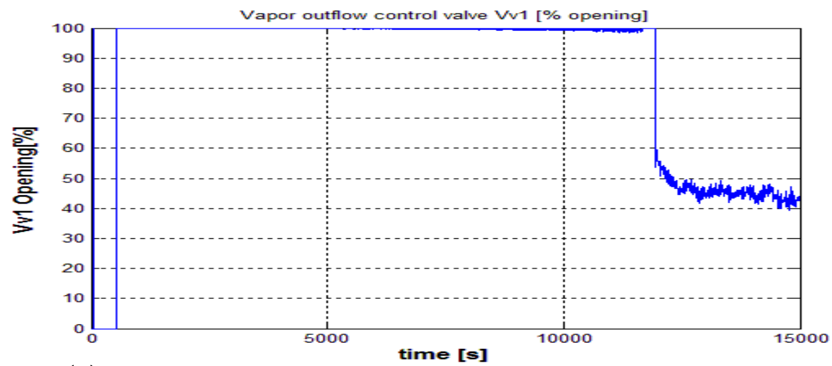


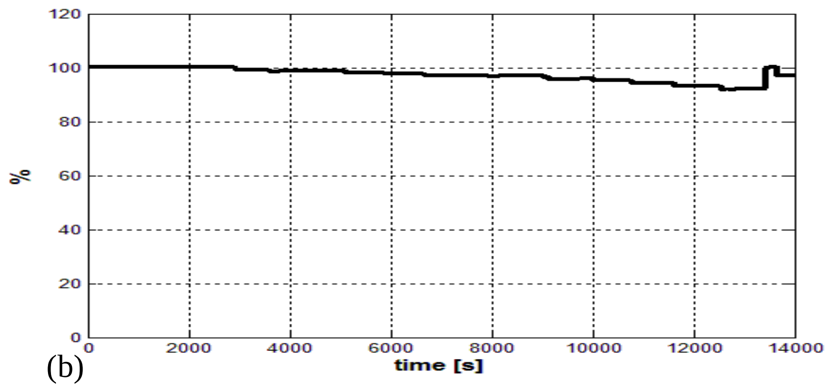
Figure 5.3 Steam inlet control valve

5.2.1.4 Vapor outflow valve Vv_1

The Figure 5.4a suggests that the vapor outlet valve should be open fully after the level reaches the target value at about 542s and then it should be operated in the range of 40%-50% opening after the product target concentration is reached at about 11921s. The control of valve Vv_1 as computed by the NLP solver approach in [1] shown in Figure 5.4b is at 100% till the appropriate product concentration is reached at about 13000s and is maintained at 98-100% even after that point. This can be understood in terms of the behavior of steam inlet valve discussed in 5.2.1.3. The opening of steam valve in [1] is assumed to be a discretely controlled valve controlled at 80% or 100% and will be controlled at 100% opening after the product concentration has reached the target value. In our case as described in Figure 5.3 the steam valve is controlled at 88-92% opening thus decreasing the heat transferred to the evaporator and hence the volume of vapor produced is also decreased significantly. Hence at steady state the valve V_{v1} needs to be at operated in the range of 40% - 50% opening.



(a)



(b)

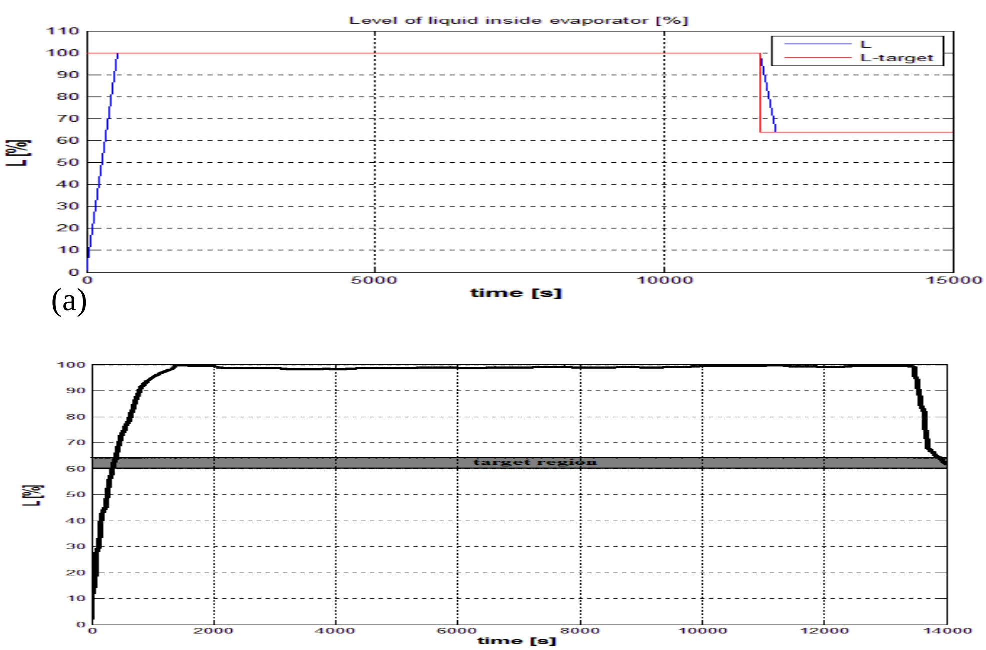
Figure 5.4 Vapor outflow control valve using a) our solver b) NLP solver in [1]

In the following sections, we discuss the dynamics of the process variables, level of liquid inside the evaporation column (L), concentration of A in product (wA), pressure of vapor inside the evaporation column (P), temperature inside the evaporation column (T) and the heat transferred to the evaporation column from the heat exchanger (Q). Alongside we compare them to the behavior of process variables in [1].

5.2.1.5 Level of liquid inside the evaporation column

The plot of level variable in Figure 5.4b is the result obtained in [1]. It can be inferred that the level inside the evaporation column rises to the value of 100% in about 1500s. The level is maintained close to 100% till about 13000s. After this time the level is reduced to the target range of 60-64% because the concentration reaches a value of target concentration minus 0.09 (0.71) at this time, as described in section 3.3.3.

The plot of level in Figure 5.4a indicates that the evaporator level rises to 100% in about 550s and is controlled at 100% until the concentration of A in product attains a value of target concentration minus 0.09, same as above at time 11663s. Thereafter the level is brought down to the target steady state value of 64%. As discussed in section 3.3.3 this is done in order to maximize the heat transferred to the evaporation column from the heat exchanger. This is similar to the desired behavior of level inside the evaporation column. However in this case the rate of level increase is marginally higher on account of the 80% opening of steam valve from beginning till the time when level reaches its maximum value. The amount of liquid forming vapor in this region is lower due to lower heat transferred from the heat exchanger. This is also indicated by the near flat region till 500s in the Figure 5.6a. As a result of this the level of liquid reaches 100% at a faster rate and also leads to overall reduction in time taken for the startup process. It is observed from Figure 5.3 that the steam valve opens to 100% after level reaches 100%. The reason for such steam valve control can be attributed to the integral quadratic performance index formulated in section 3.3.3.



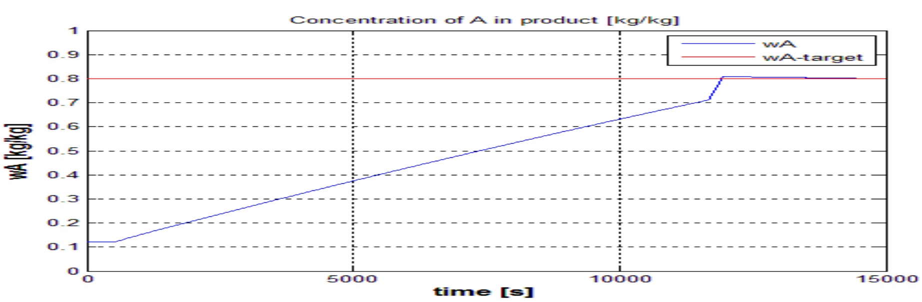
(a)

(b)

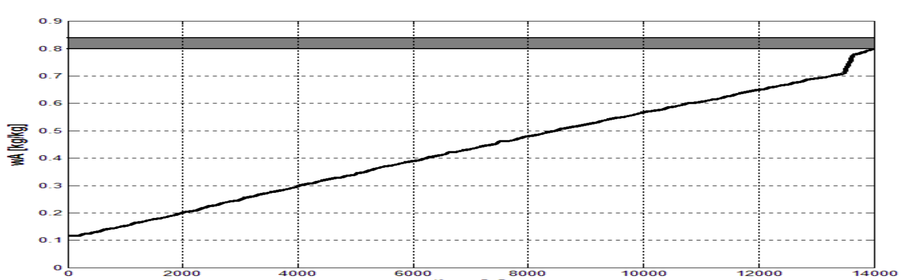
Figure 5.5 Level of liquid in the evaporator using a) our solver b) NLP solver in [1]

5.2.1.6 Concentration of A in product

Figure 5.6b depicts the rise in concentration of component A in product from the starting value of 0.12 to the target value of 0.8 as obtained from [1]. It is observed that at about 13000s the concentration of A in product suddenly rises to 0.8. Figure 5.6a also depicts the rise in concentration of component A to the target value of 0.8. The flat portion at the beginning of the curve indicates the region where the steam valve is controlled at 80% opening and hence the rate of evaporation is low. As a result the change in concentration of A is very low. However the concentration starts rising at around 550s when the level attains maximum value, as the steam valve opens at that time thus causing more evaporation of the solvents. The steep rise in concentration towards the end around 11663s is also observed in this figure as pointed out earlier in Figure 5.6b. This can be attributed to the change in weights of the cost function when the concentration reaches the value of 0.71 (0.8 – 0.09) as described in section 3.3.3. The level is brought to the steady state value of 64% during this jump. This can be seen as flash-evaporation of the liquid caused due to the level reduction thus causing more liquid to evaporate in a very short span of time. This causes the concentration of A to rise sharply and reach the target value of 0.8.



(a)

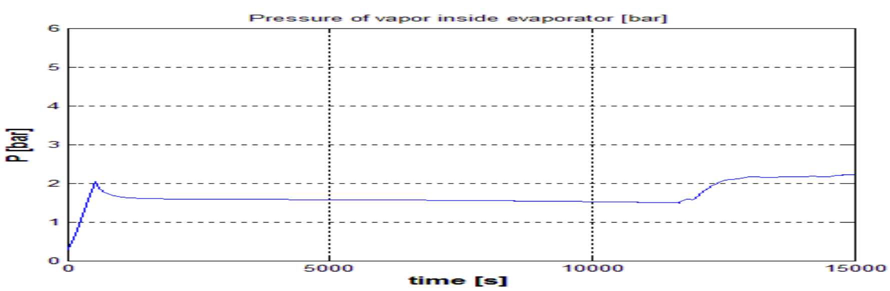


(b)

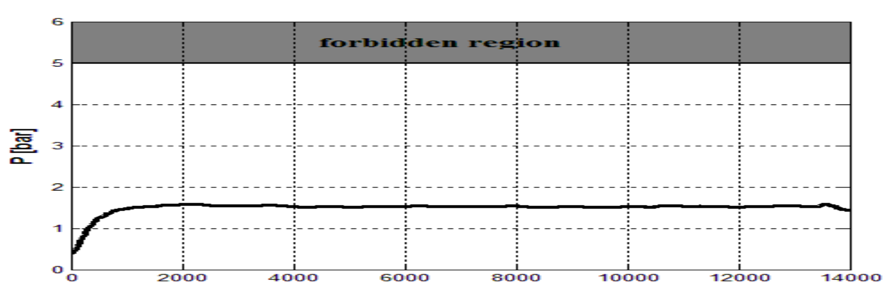
Figure 5.6 Concentration of A in product using a) our solver b) NLP solver in [1]

5.2.1.7 Pressure of vapor inside the evaporation column

From the plot of evaporator pressure in Figure 5.7a it is observed that the pressure inside the evaporator is maintained close to 1.6 bar and sharply rises to around 2.4 bar after the product has reached the target concentration of component A at around 11921s. This is different from the variation of pressure obtained in [1] depicted in Figure 5.6b only towards the end of the process where the pressure appears to dip. However the data after 14000s is not available in these results. The steep rise in the pressure towards the end in our case is attributed to the rise in temperature explained clearly in the next section. Also, the rate at which the product is dispatched is relatively lower in our case as the opening % of product outflow valve in our case is about 8% at the end of the process as compared to 11.5% considered in [1]. This rise in pressure can hence be intuitively justified using the relationship $P \propto T/V$ from the ideal gas law. It is possible that the volume of the vapor increases at a lower rate than the rise in temperature inside the evaporation column at that time, thus causing net increase in pressure towards the end. However it is seen that the pressure is maintained below the safe range of 5bar pressure throughout the control process and hence the safety violation constraints are never activated.



(a)

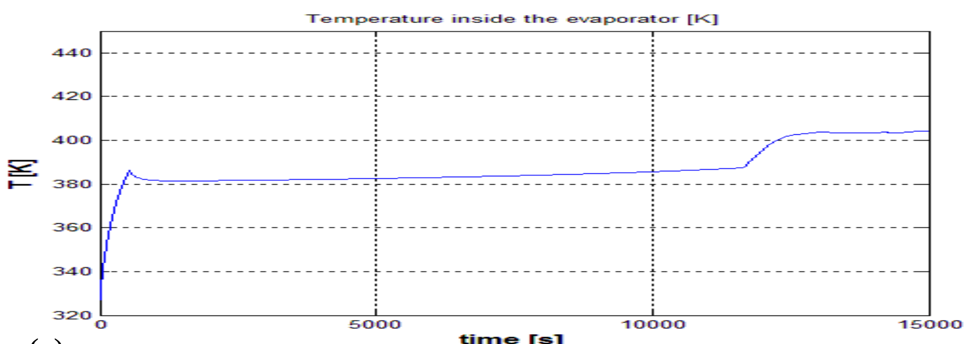


(b)

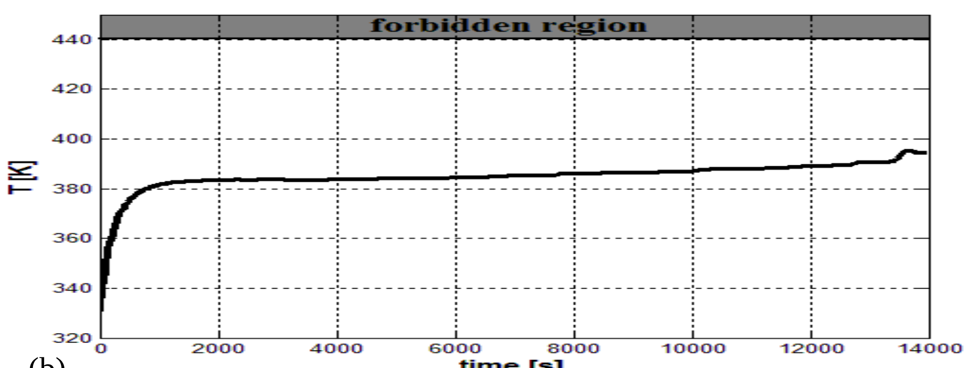
Figure 5.7 Pressure of vapor in evaporator using a) our solver b) NLP solver in [1]

5.2.1.8 Temperature inside the evaporation column

Figure 5.8a shows the temperature variation inside the evaporation process during the startup process. It is seen that the temperature rises to around 385K in the beginning (0-550s) and then increases at a steady rate to a value of 392K at the time when product reaches the target concentration (~600s to 11663). It is observed that the temperature steeply rises thereafter to a value of around 407K. Rise in temperature towards the end of the process is also observed in the plot obtained in [1] using the NLP solver (Figure 5.8b). This occurs possibly because the level drops to a value of 64% after the target product concentration is attained. As a result of this less amount of liquid is heated by the heat transferred from the heat exchanger and the temperature rises rapidly. The profile of the temperature curve is slightly different due to the minor differences in the performance index used in our approach as compared to [1] as discussed in section 3.3.3. However the initial temperature and temperature value at the time the product attains the target concentration are identical in both the cases.



(a)

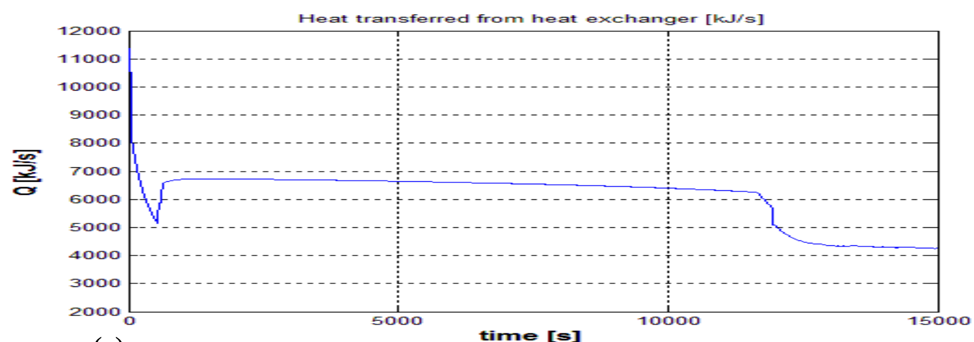


(b)

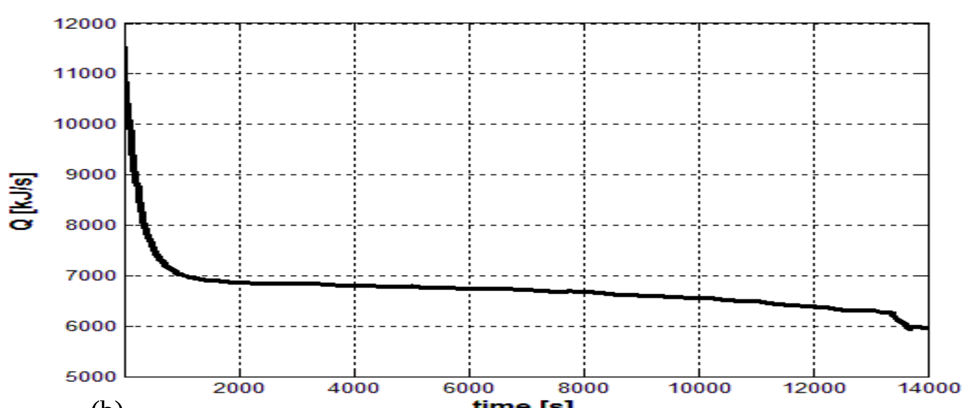
Figure 5.8 Temperature inside the evaporator using a) our solver b) NLP solver in [1]

5.2.1.9 Heat transferred to the evaporator from the heat-exchanger

The heat transferred to the evaporator decreases rapidly after the product attains the target concentration as apparent at about 12000s in Figure 5.9a and at about 13500s in Figure .9b. This behavior is attributed to two reasons; firstly the sharp rise in evaporator temperature as explained in the previous section reduces the temperature difference between the heat exchanger and the evaporation column. This leads to reduction in heat transfer. Secondly, in our case the steam valve opening is reduced from 100% to about 86% at this time and hence the dip in the heat available from steam condensation. We also observe a discrete jump in the heat transferred from the heat exchanger in Figure 5.9a at the beginning (~550s). The time at which this occurs is same as the time when the steam valve opens as shown in Figure 5.3. This is justified as there is a sudden rise in the amount of steam entering the heat exchanger at that time. The profile of the heat transferred is very similar to the plot obtained from the results of [1].



(a)



(b)

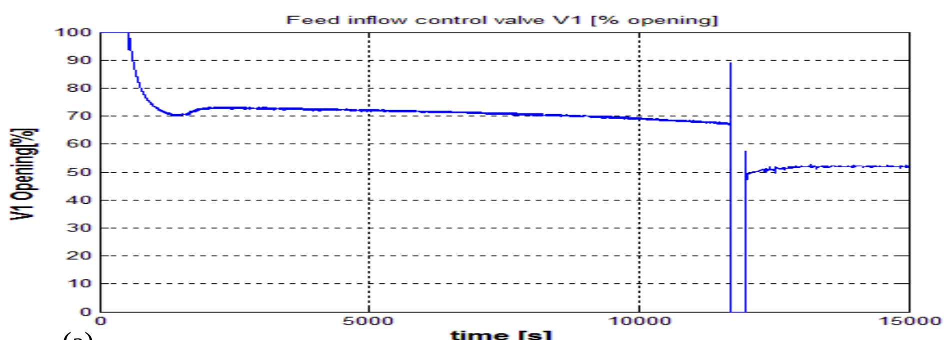
Figure 5.9 Heat transferred from heat-exchanger using a) our solver b) NLP solver in [1]

5.2.2 Case 2: Two-partition-window NMPC, $h_{opt} = 1$ results

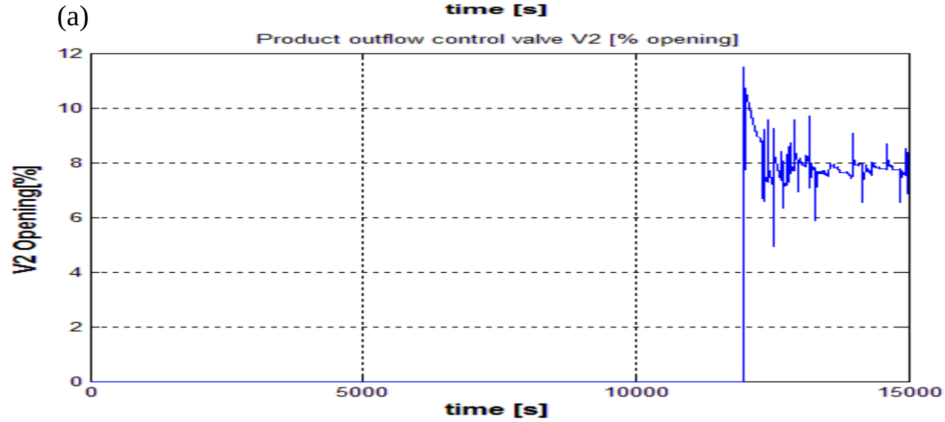
In this section we discuss the results of the two-partition-window NMPC approach with $h_{opt} = 1$. The behavior of the process variables is exactly identical to the plots described in the previous section however we observe a noticeable improvement in the sequence of input controls computed by the solver as discussed in the following sections.

5.2.2.1 Control of inflow and outflow valves

Due to the higher resolution of computation we observe that the controls obtained are finer and more accurate compared to the input and output valve controls depicted in section 5.2.1. Specifically it can be seen from the plot in Figure 5.10 a and Figure 5.11 a, b that the input valve controls governing the opening of input feed valve, the vapor outflow valve and the steam input valve are smoother after the product has reached the desired concentration level (at about 11921s), as compared to the plots in Figure 5.1, Figure 5.2 and Figure 5.3 respectively.

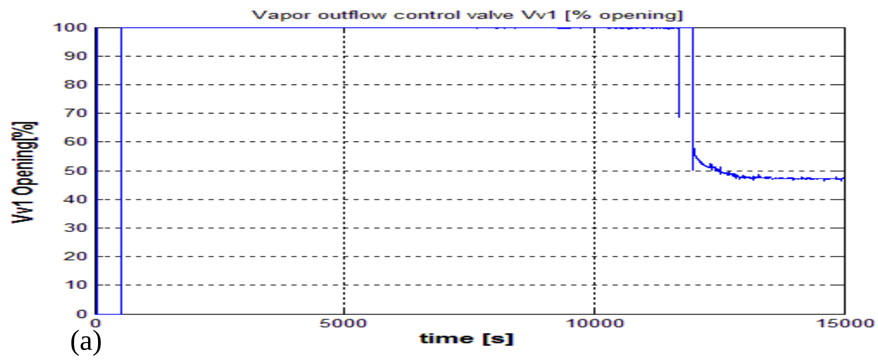


(a)

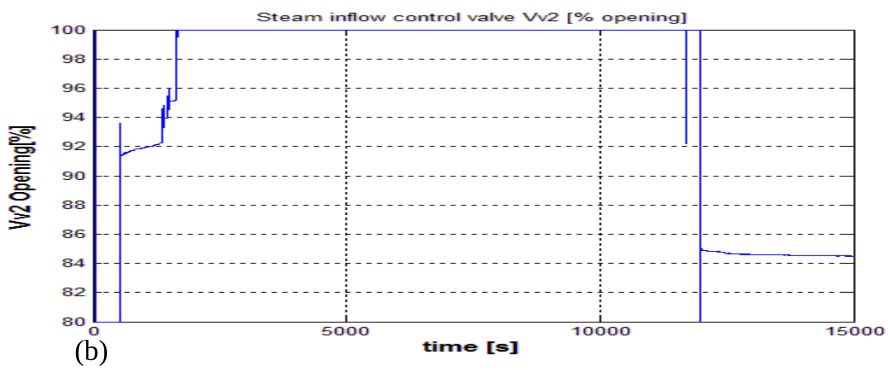


(b)

Figure 5.10 a) Feed inlet and b) product outflow valves (two-partition-window), $h_{opt} = 1$



(a)



(b)

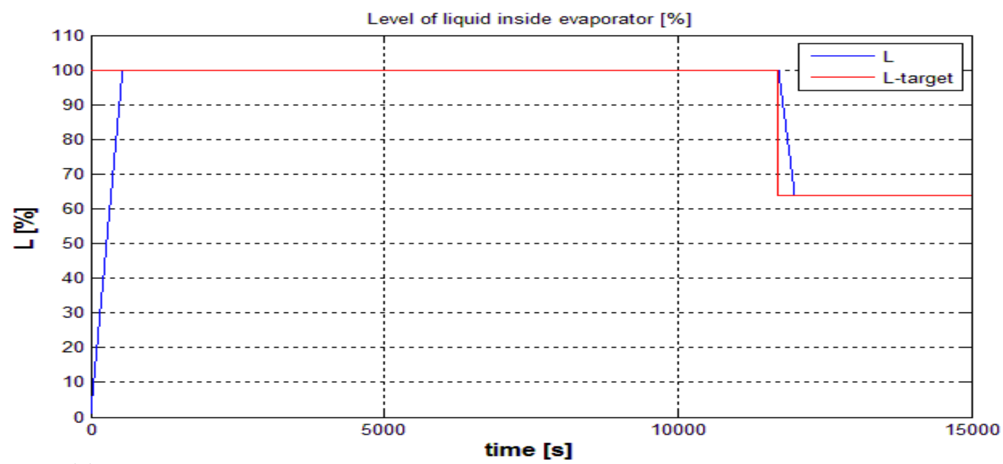
Figure 5.11 a) Vapor outflow valve and b) steam valve (two-partition-window), $h_{opt} = 1$

The plot in Figure 5.10b indicates a similar average value of 8% as in figure 5.2 and is relatively less noisy. It is important to note that, the two-partition-window case requires the solver to compute twice the number of inputs per time step and hence takes more time to converge. However due to more accuracy in simulation and more data on error from the future time-steps the two-partition window gives less noisy and more accurate controls.

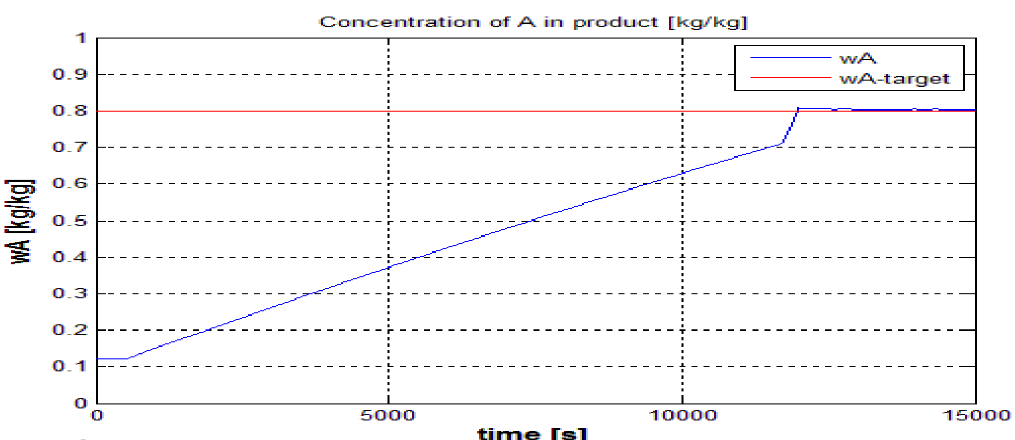
Also, we can clearly observe the difference in computation time for convergence using this approach as compared to the computation time using single-window approach. This approach takes about triple the time required for single window approach. The differences in computation times are clearly shown in Table 5.2.

5.2.2.2 Level of liquid and product concentration inside the evaporator

Next, we discuss the plots of dynamics of the output variables level and product concentration obtained from this approach. The level inside the evaporation column varies exactly similar to the level in the previous case. This is clearly depicted by the plots shown below in Figure 5.5. The evaporator level is controlled at 100% level till the desired product concentration is near the target value at around 11663s and is brought down to the target range of 64% after the concentration of A in product reaches a value of 0.71 (0.8 -0.09).



(a)



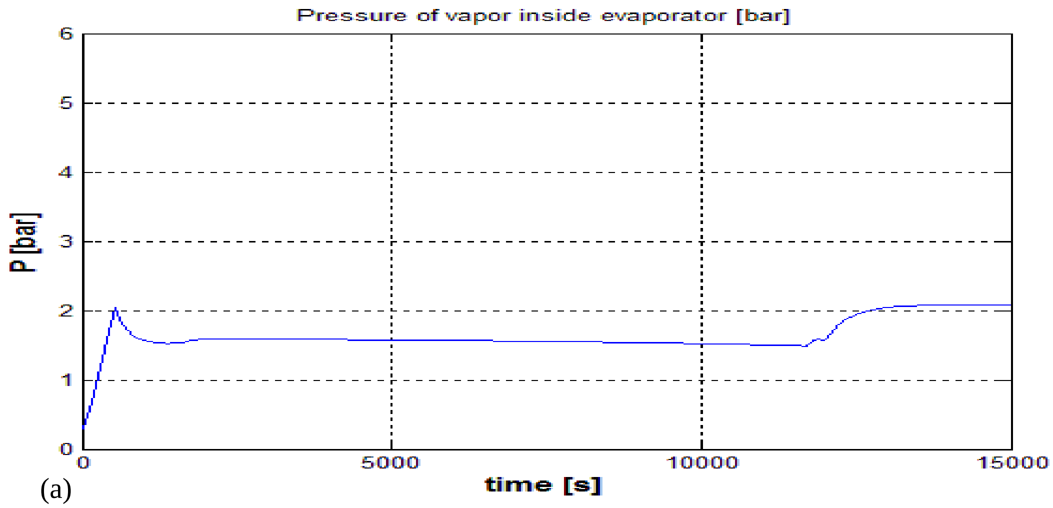
(b)

Figure 5.12 a) Level and b) product concentration (two-partition-window, $h_{opt} = 1$)

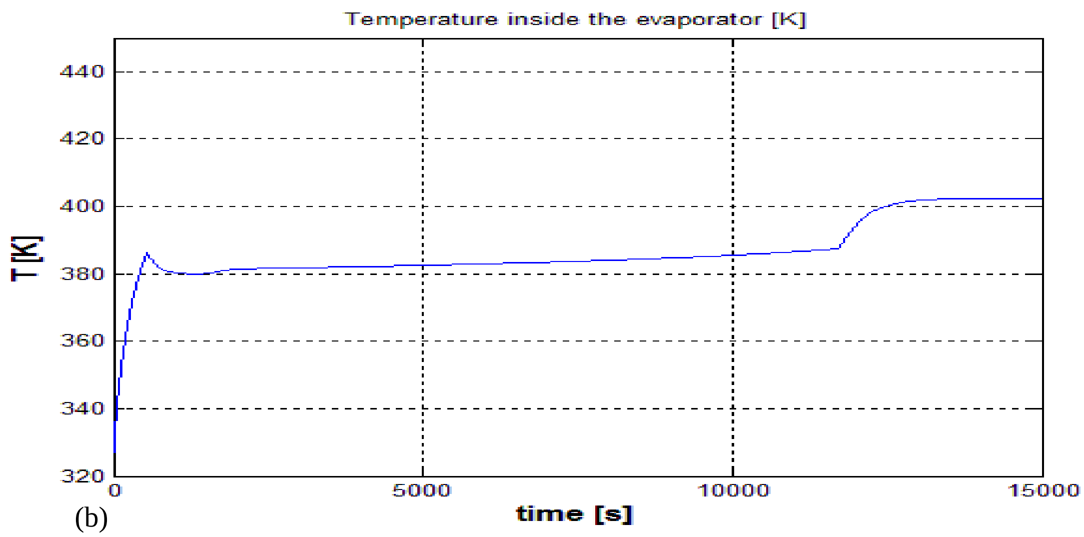
The product concentration depicted in Figure 5.12b is quite similar to the behavior observed in Figure 5.6a and b.

5.2.2.3 Pressure and temperature inside the evaporator

Similar to the discussion in section 5.2.1.7 the pressure inside the evaporator sharply increases at the end of the startup process. Again this can be attributed to the same reason as discussed in section 5.2.1.7. It can be seen from the Figure 5.12a that the pressure variation is similar to the plot in Figure 5.6a and compares well with the pressure obtained in [1] during the startup process as depicted in Figure 5.6b. Also the pressure inside the evaporator is maintained below the safe range of 5bar pressure throughout the startup process.



(a)



(b)

Figure 5.13 a) Pressure and b) Temperature inside the evaporator (two-partition-window, $h_{opt} = 1$)

The variation of temperature inside the evaporator is again very similar to the behavior observed in Figure 5.7a. As explained in that case, the sudden rise in temperature towards the end of the startup process is attributed to the reduction in level as a result of which less volume of liquid is heated by the steam flowing into the heat exchanger.

5.2.2.4 Heat transferred to the evaporator from the heat-exchanger

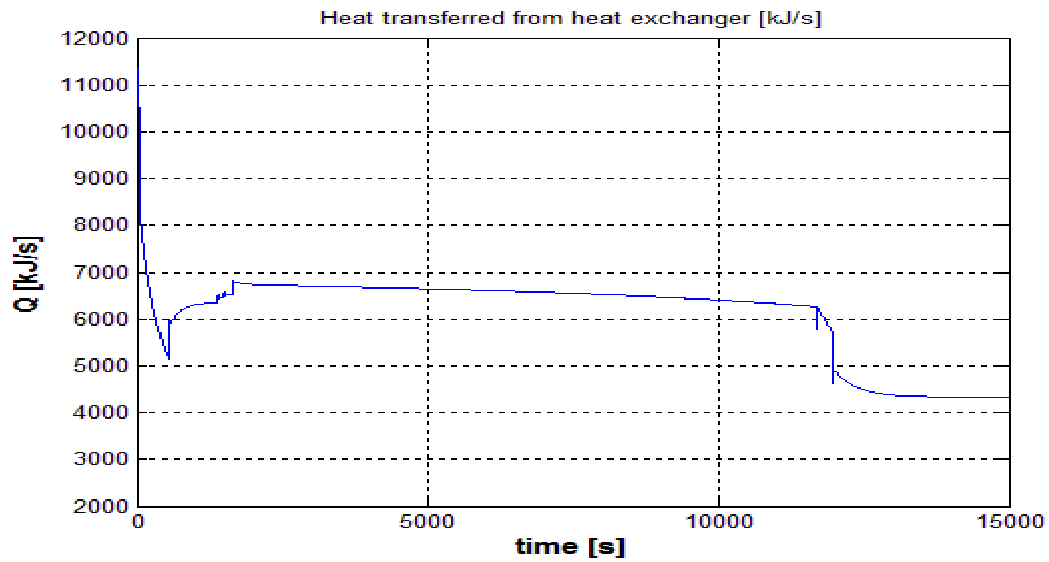


Figure 5.14 Heat transfer from the heat exchanger (two-partition-window, $h_{opt} = 1$)

Similar to the argument in section 5.2.1.9, there is a sudden jump in the heat transferred at the beginning of the process caused by sudden opening of the steam inflow valve. The dip in heat transfer towards the end of the startup process is again attributed to the same reason discussed in section 5.2.1.9.

Hence, we come to the end of this section and conclude that apart from the finer control values obtained in this case, all the other process variables had very similar behavior as compared to the process variables discussed in section 5.2.1. Also it will be seen in section 5.3 later that the execution time for this approach takes about triple the time taken by the single-window 1s prediction horizon case.

5.2.3 Case 3: Two-partition-window NMPC, $h_{opt} = 0.5$ results

In this section we discuss the results of the two-partition-window NMPC approach with $h_{opt} = 0.5$. The behavior of the process variables is identical to the previous two sections however we again observe that we get even better results for the sequence of inputs computed by the solver. The controls obtained are finer and more accurate compared to the input and output valve controls depicted in Figure 5.10 and Figure 5.11.

5.2.3.1 Control of inflow and outflow valves

Due to the even more resolution as compared to the previous case, we see from the Figure 5.15 and 5.16 that the controls obtained are finer and less noisy as compared to the input and output valve controls depicted in section 5.2.2. The control sequence interpreted by the plots in these figures is also similar to the ones obtained in section 5.2.1.

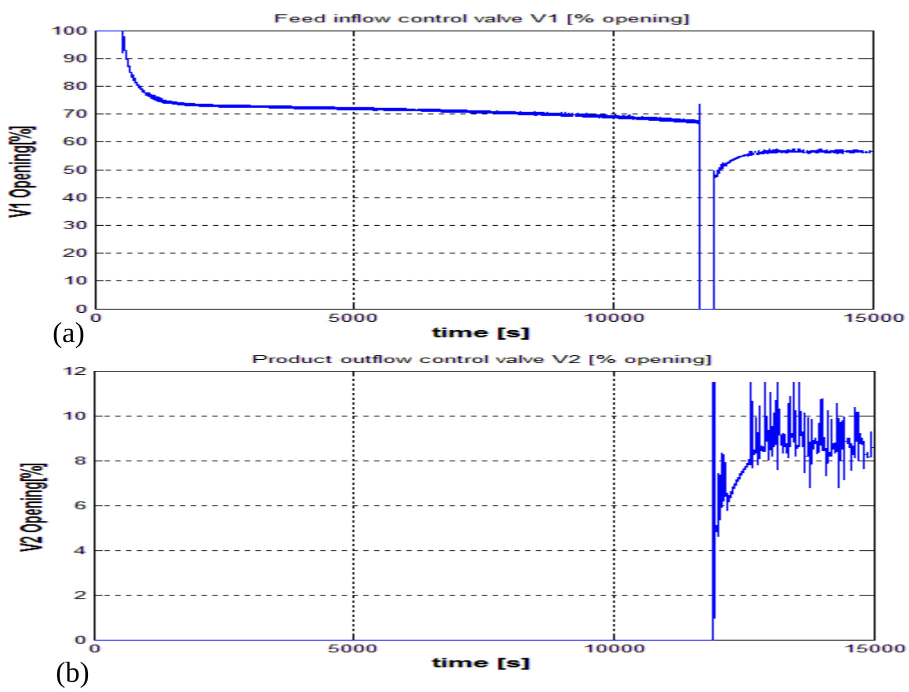
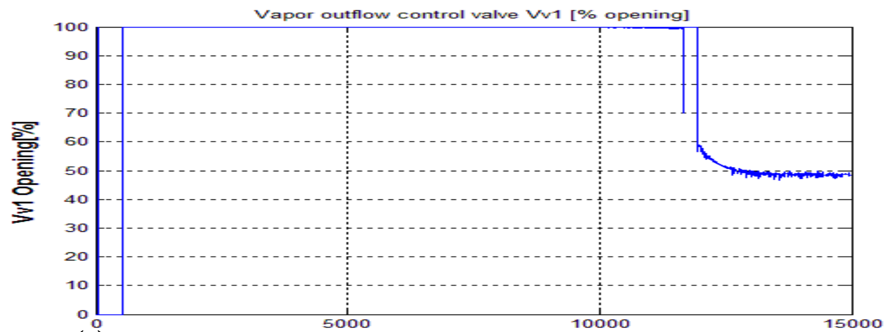
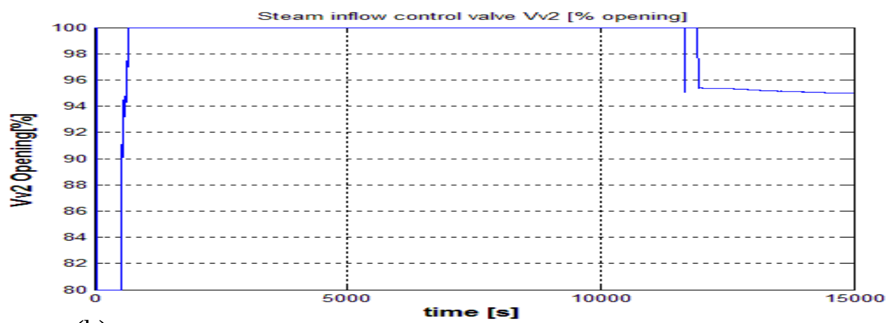


Figure 5.15 a) Feed inlet and b) product outflow valves
(two-partition-window, $h_{opt} = 0.5$)



(a)



(b)

Figure 5.16 a) Vapor outflow valve and b) steam valve
(two-partition-window, $h_{opt} = 0.5$)

We observe from Figure 5.16b that the steam valve has finer control at the beginning of the startup process. This however has minimal impact on other process variables as there is still a discrete jump in the opening of the steam valve at that time similar to the plots in Figure 5.3 and 5.11. We also observe that there is a vast difference in computation time for convergence using this approach. This is due to the fact that the number of computations in this approach is about 4 times more than the case in section 5.2.1 and twice that of section 5.2.2. As a results, this approach takes about 4 times the time required for single window approach. The differences in computation times are shown in Table 5.2 in section 5.3.

5.2.3.2 Level of liquid and product concentration inside the evaporator

In this section we see the variation of level inside the evaporation column. The profile is exactly similar to the previous case. This is clearly depicted by the plots shown below in Figure 5.17a. The evaporator level is controlled at 100% level till the desired product concentration is near the target value (0.71) at around 11663s and is brought down to the target range of 64% after the concentration of A in product reaches a value of 0.8. The product concentration also has an identical behavior as depicted in Figure 5.17b as compared to the plots obtained in Figure 5.12b.

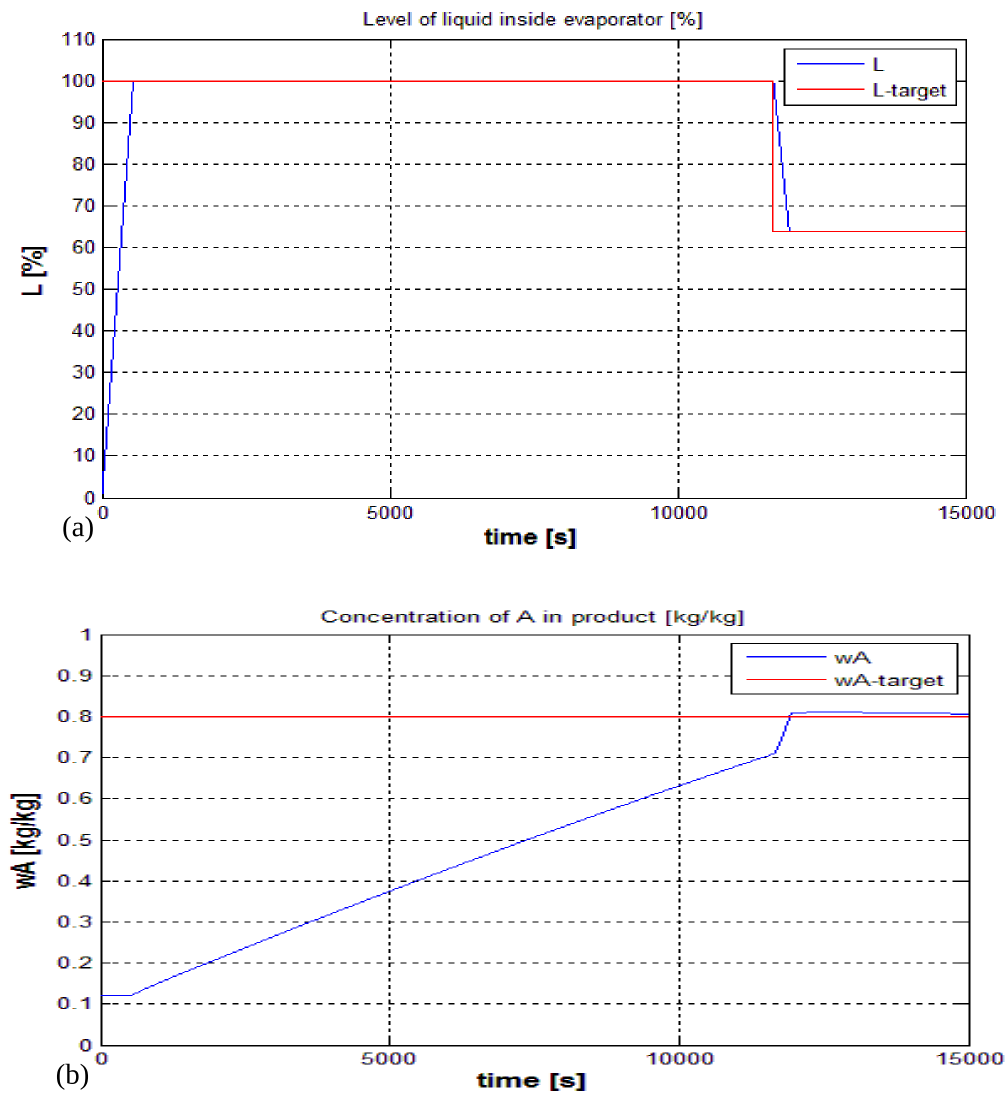


Figure 5.17a) Level and b) product concentration (two-partition-window, $h_{opt} = 0.5$)

5.2.3.3 Pressure and temperature inside the evaporator

The behavour of pressure and temperature for this case depicted in Figure 5.18 a and b has the same profile as the plots in Figure 5.13. The reason for the sharp rise in values at the end of the process is same as discussed in the previous section.

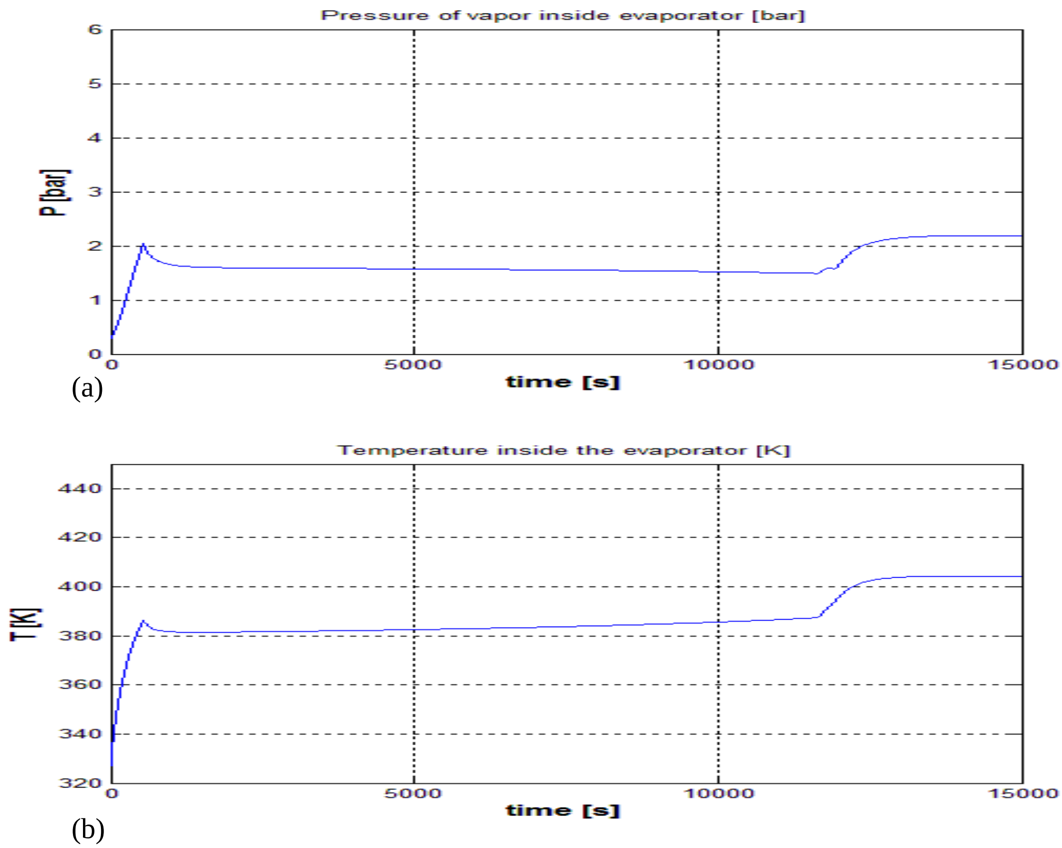


Figure 5.18 a) Pressure and b) Temperature inside the evaporator (two-partition-window, $h_{opt} = 0.5$)

5.2.3.4 Heat transferred to the evaporator from the heat-exchanger

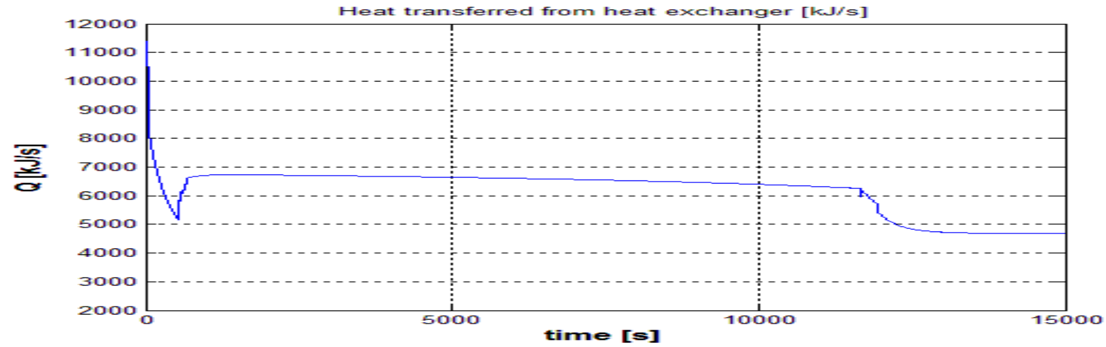


Figure 5.19 Heat transfer from the heat exchanger (two-partition-window, $h_{opt} = 0.5$)

The curve of heat transfer as depicted in Figure 5.19 is marginally different in this case due to the marginal differences in the steam valve control as discussed in section 5.2.3.1. But the overall profile of the heat transfer remains same as those presented in section 5.2.2.4.

It is important to note that, the two-partition-window with $h_{opt} = 0.5$ case requires the solver to compute twice the number of inputs per time step and has double the number of time steps compared to the earlier cases and hence takes lot more time to converge. However with more accuracy in simulation and more data on error from the future time steps the two-partition window with h_{opt}=0.5 gives less noisy and more accurate controls compared to the previous cases.

5.2.4 Two-partition-window NMPC, $h_{opt} = 0.5$ results (superheated steam)

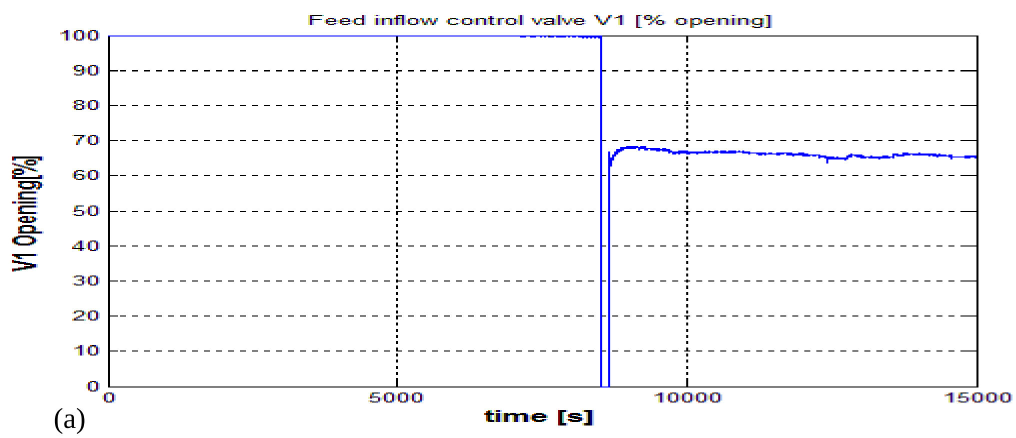
When solving for the startup of evaporation system using a single evaporation column we identified that the pressure (5bar) and temperature (440K) of the steam entering the heat exchanger specified in [2] for this case correspond to super-heated steam. The system solved in [1] is extension of the work by C. Sonntag in [2]. Thus the parameters estimated for the thermodynamic model of the specific heat capacity of incoming vapor as given in Appendix B are considered as:

$$c_{p,v}(\xi) = 0.0019325\xi_B + 0.0016115\xi_C$$

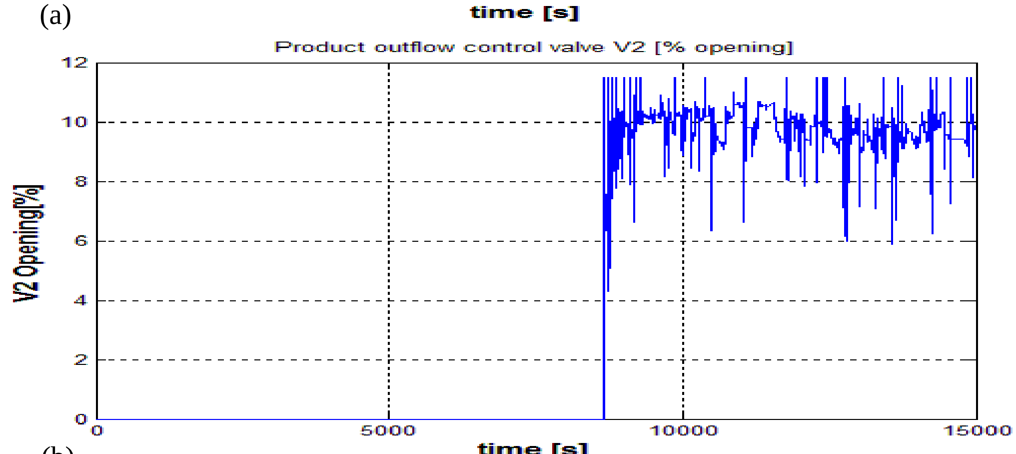
However the specific heat capacity of superheated steam at the specified temperature and pressure condition was found to be 2.31625 kJ/Kg/K. Hence in this section we present the results obtained by using this value for the specific heat capacity of the superheated steam flowing into the heat exchanger. Following are the inflow and outflow control inputs and the results of the process variable values obtained.

5.2.4.1 Control of inflow and outflow valves

The control sequence evident from the plots in Figure 5.20 a and b, is to open the feed inlet valve V_1 till around 8000s for the product to attain target concentration and then maintain V_1 at 61-63% opening in the steady state of operation. The product out flow valve is closed till the product reaches target concentration and is opened at 10-11% opening during dispatch of the product in the steady state.



(a)



(b)

Figure 5.20 a) Feed inlet and b) product outflow valves (two-partition-window, $h_{opt} = 0.5$) for the super-heated steam case.

The plot in Figure 5.21 a, b depict control of steam inflow valve, Vv_2 and vapor outflow valve, Vv_1 respectively. In this case the steam valve is not opened at 100% for an efficient startup, unlike the cases considered so far. This is due to the higher heat capacity of steam considered in this case. The heat transferred to the system per unit mass of steam will be higher than all the previous cases. Hence the steam valve needs to be controlled at 80% opening till about 8000s and then maintained at about 88-89% opening after the product concentration has been attained. The profile in Figure 5.21a in case super-heated steam is used as the heating fluid. On the other hand Figure 5.21b gives a profile for controlling the vapor outflow valve opening. The point where both these curves show the momentary impulsive jump in both these plots is the time when the target product concentration is attained.

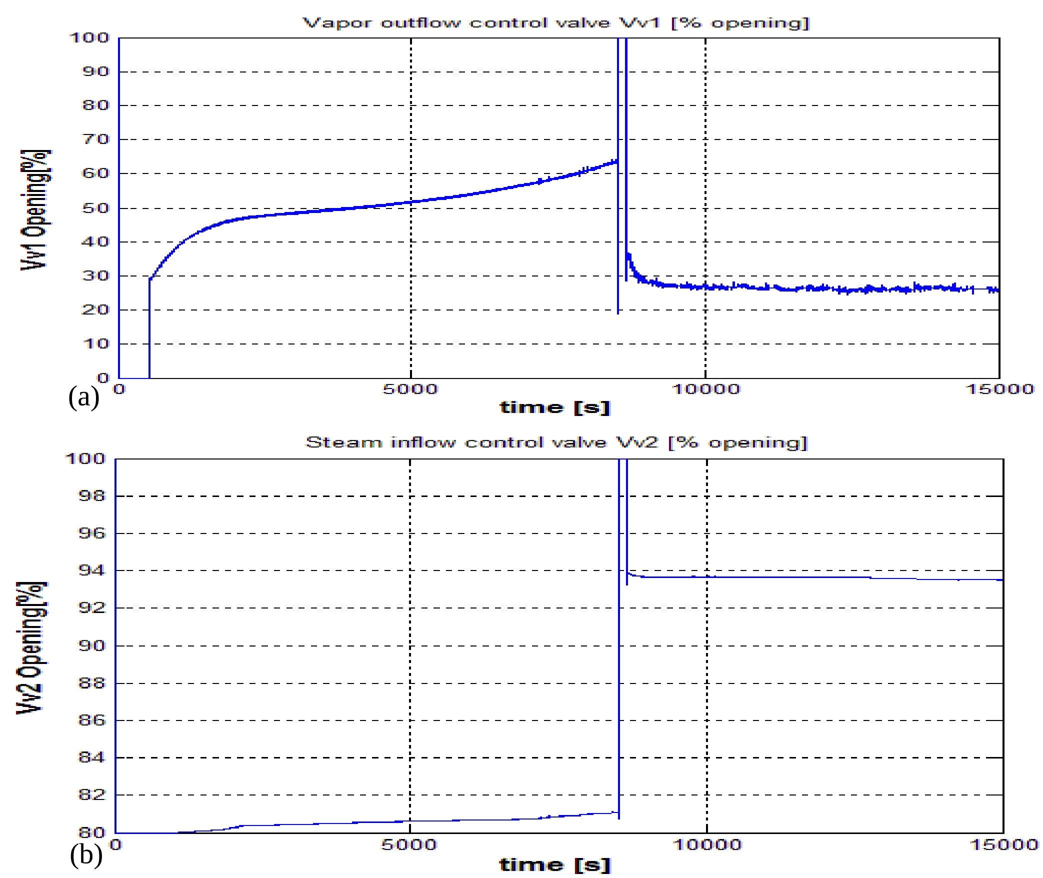
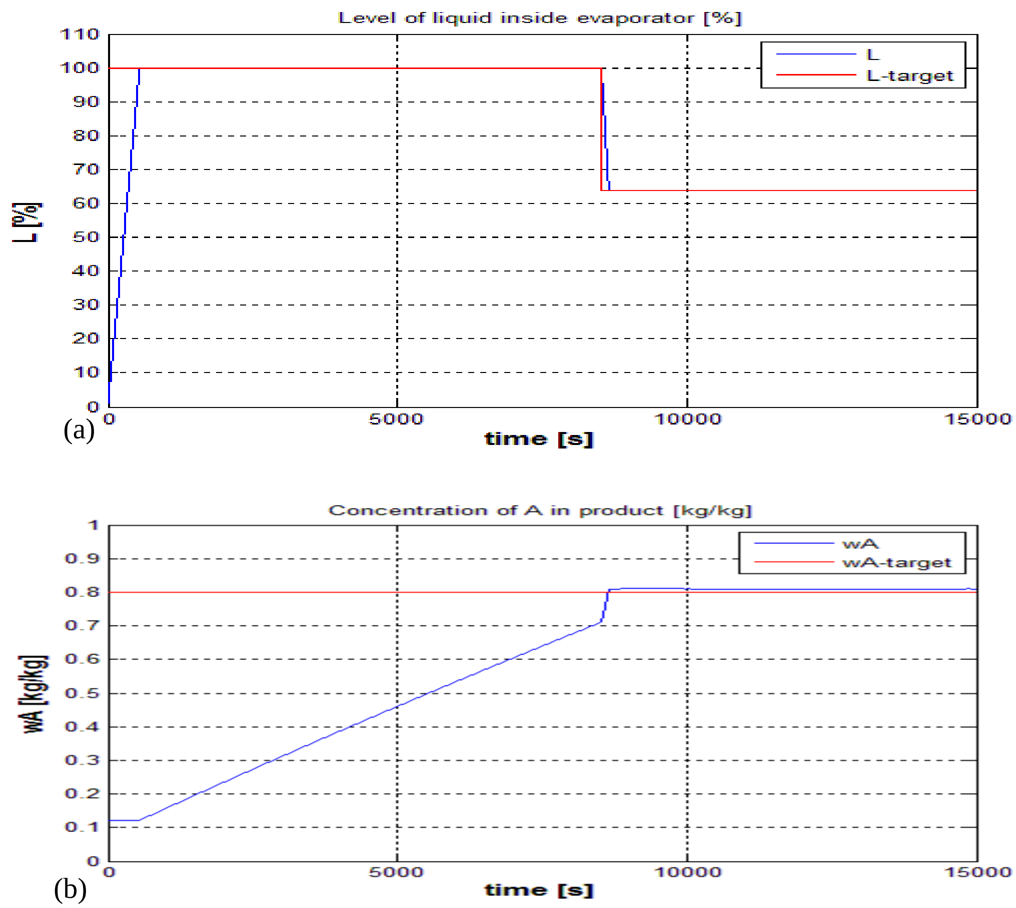


Figure 5.21 a) Vapor outflow valve and b) steam valve (two-partition-window, $h_{opt} = 0.5$) for the super-heated steam case.

5.2.4.2 Level of liquid and product concentration inside the evaporator

Figure 5.22 shows a good level tracking in this case. The level is maintained at 100% till the product attains target concentration and then drops to 64% opening in steady state which is desirable. It is observed From 5.22b that the concentration of component A in product rises at a relatively faster rate in this case as more heat is transferred from the heat exchanger per unit time.



(a)
(b)

Figure 5.22 a) Level and b) product concentration (two-partition-window, $h_{opt} = 0.5$) for the super-heated steam case.

5.2.4.3 Pressure and temperature inside the evaporator

Figure 5.23a shows that the variation of pressure is beyond the 5bar value and hence this case is feasible only when the vessel is designed to handle a pressure of upto 7bar. The temperature inside the evaporator according to Figure 5.23b varies within the maximum limits of 440K. Temperature is maintained in the range of [410 K, 430K].

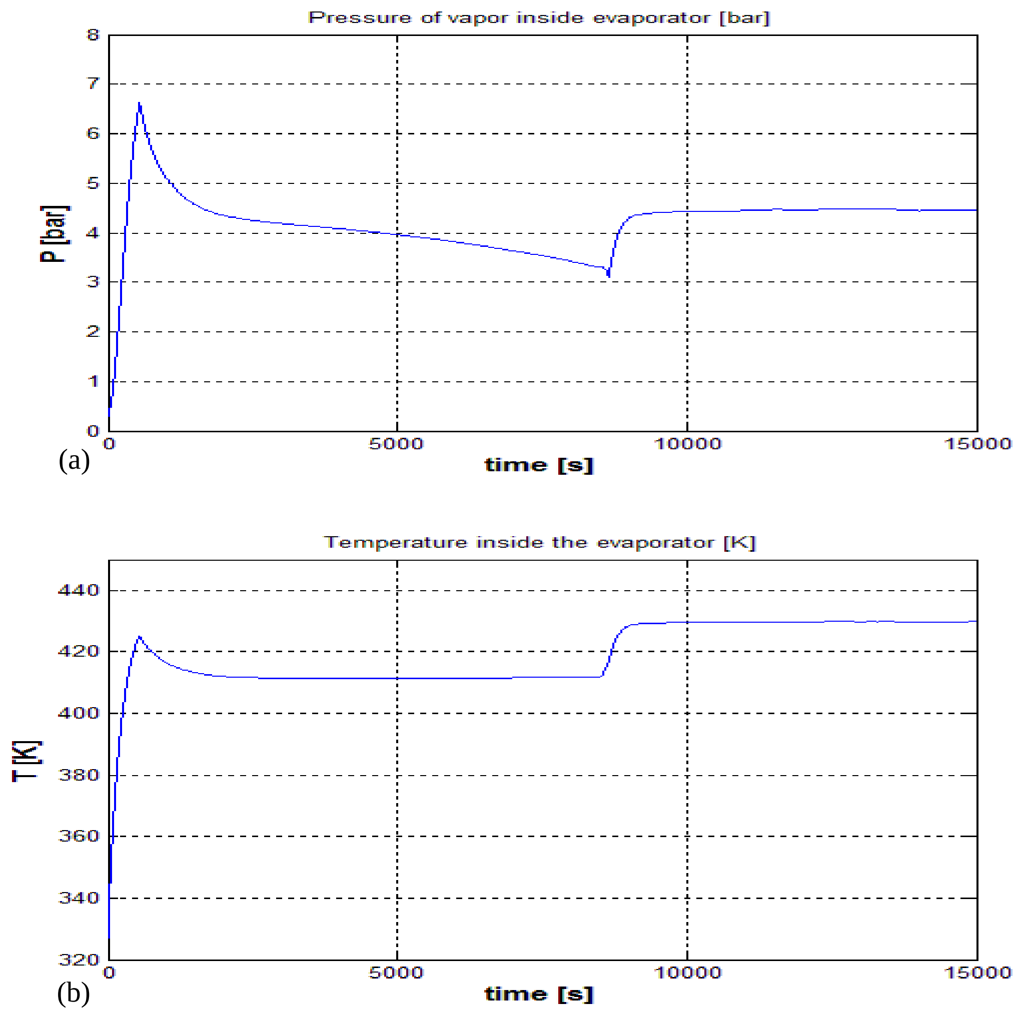


Figure 5.23 a) Pressure and b) Temperature inside the evaporator (two-partition-window, $h_{opt} = 0.5$) for the super-heated steam case.

5.2.4.4 Heat transferred to the evaporator from the heat-exchanger

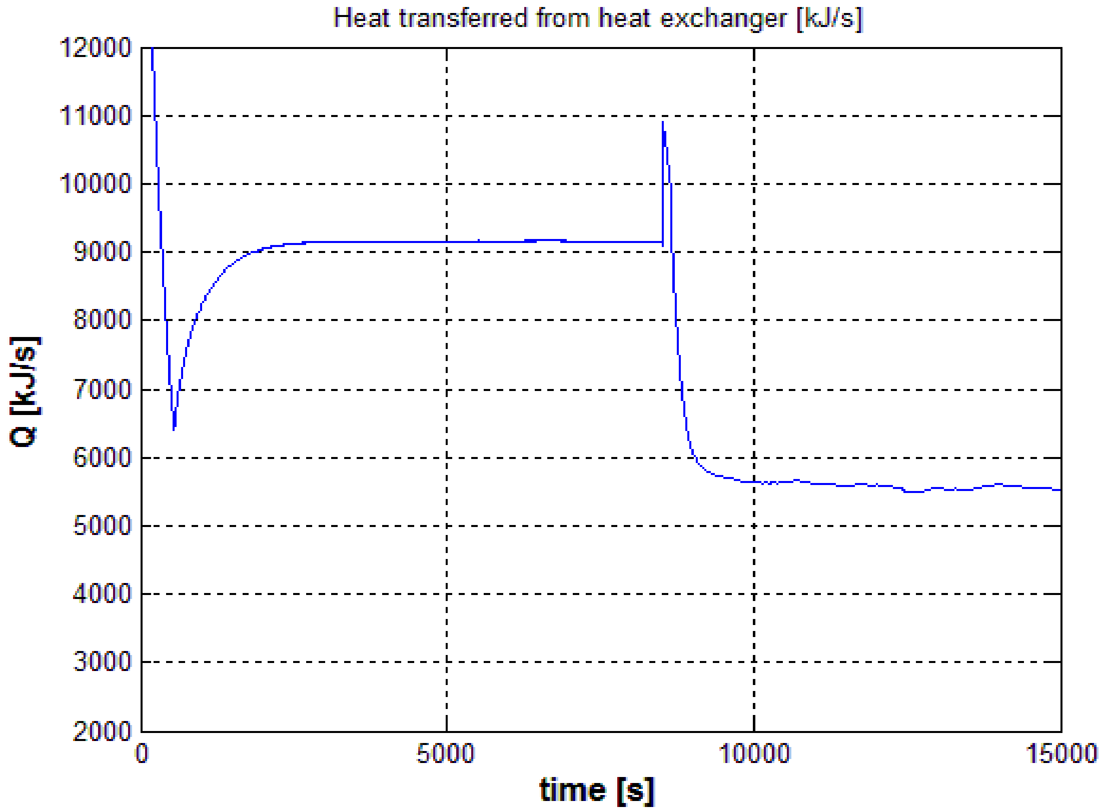


Figure 5.24 Heat transfer from the heat exchanger (two-partition-window, $h_{opt} = 0.5$) for the super-heated steam case.

The profile for the heat transferred from the heat exchanger is very identical to the other cases considered in this chapter. However due to the difference in heat capacity of fresh steam the value of heat transferred from the heat exchanger is relatively higher s compared to the other cases. The program takes about 48478s which is about 13.5hours to converge for this case. The controls calculated are very smooth and accurate.

5.2.5 Single -window NMPC, $h_{opt} = 100$, $h_{sim} = 1$ results

In this section we discuss the results of the single-window NMPC approach with $h_{opt} = 100s$. This essentially means that the inputs are assumed to be constant for a minimum of 100s and the solver solves for an optimal input change after every 100 seconds. Being a slow process this is determined to be a better combination of h_{opt} and h_{sim} values as the control inputs do not oscillate and lead to a stable control of the process. The behavior of the process variables is very similar to the previous two sections however we again observe that we get even better results for the sequence of inputs computed by the solver. The controls obtained are stable and oscillate less compared to the input and output valve controls depicted in the previous sections.

5.2.5.1 Control of inflow and outflow valves

Due to larger prediction window in this case we see from the Figure 5.25 and 5.26 that the controls obtained are less noisy as compared to the input and output valve controls depicted in the previous sections. The control sequence interpreted by the plots in these figures is also similar to the ones obtained in section 5.2.1. From Figure 5.25a and Figure 5.25b it is observed that at steady state, the inlet feed valve and product outflow valves are open at about 58% and 8.5% respectively.

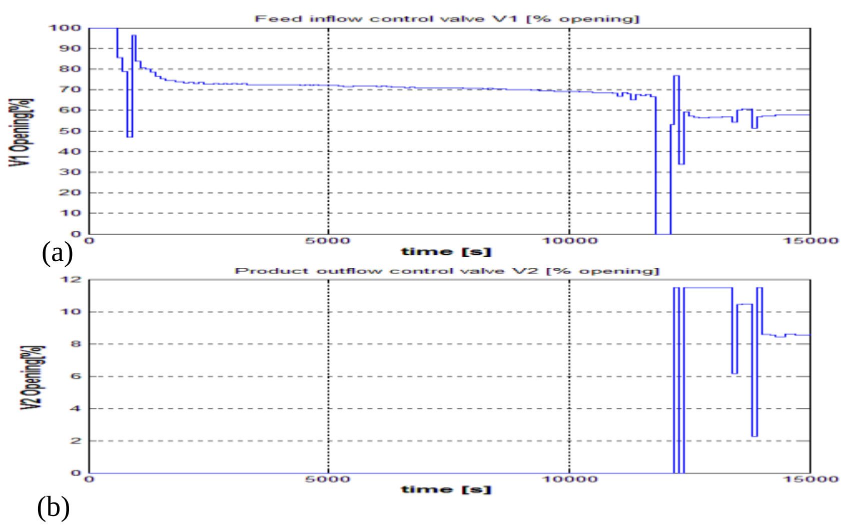


Figure 5.25 a) Feed inlet and b) product outflow valves (single-window, $h_{opt} = 100, h_{sim} = 1$)

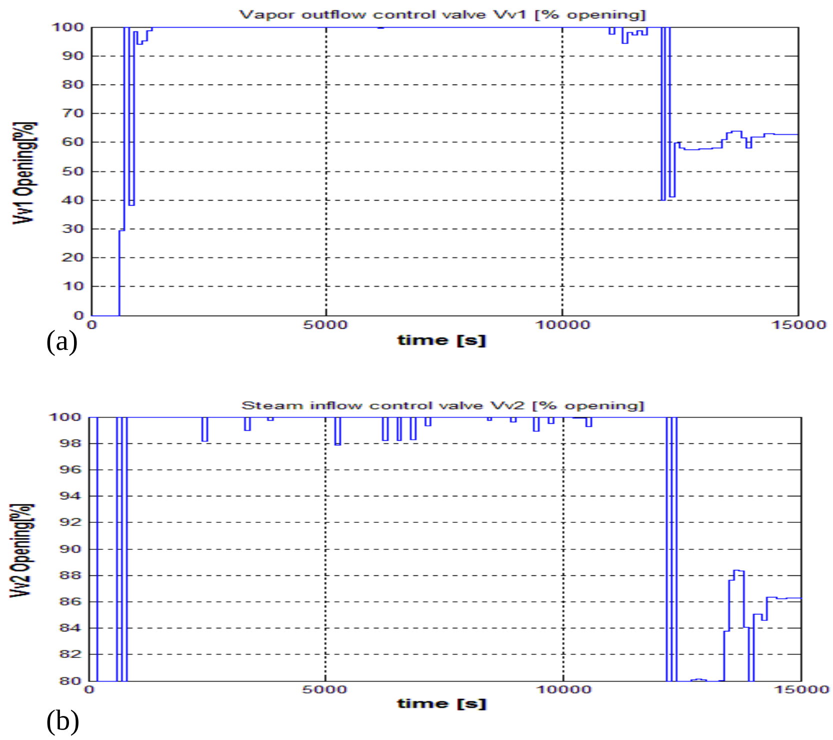


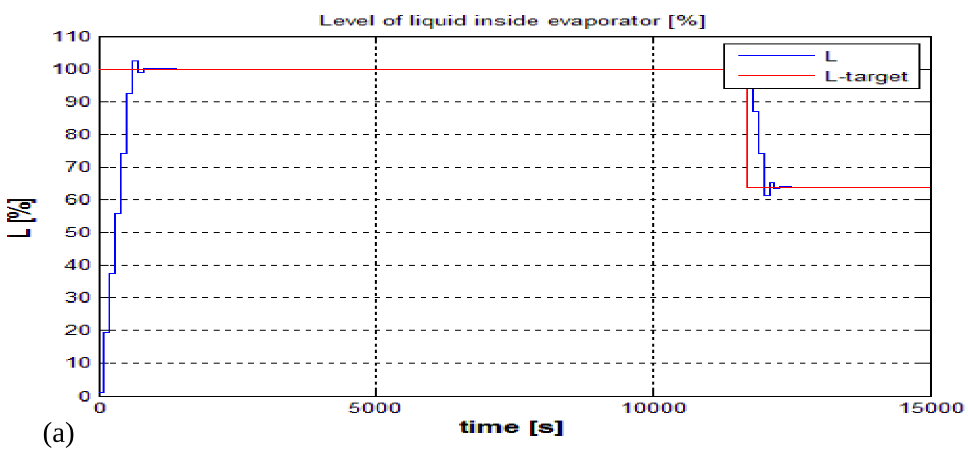
Figure 5.26 a) Vapor outflow valve and b) steam valve (single-window, $h_{opt} = 100, h_{sim} = 1$)

From Figure 5.26a, the vapor outflow valve is opened at around 550s and is kept open until about 11921s when the concentration attains the target concentration. The vapor valve is then opened at a steady state range of 58% - 65% opening. This is similar to the sequence of operation of the vapor valve as discussed in section 5.2.1 and 5.2.2.

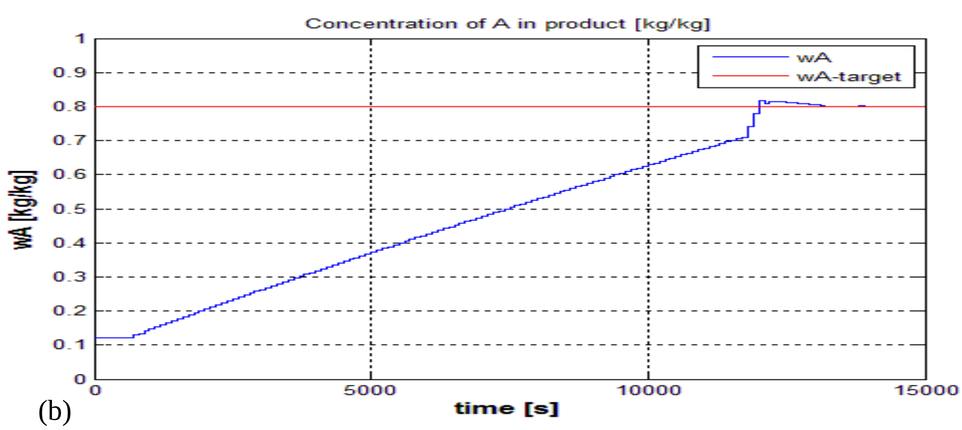
We observe from Figure 5.26b that the steam valve is operated at 80% opening until the level inside the evaporation column attains a maximum value at nearly 550s similar to the cases discussed thus far. Thereafter the steam valve is opened at 100% opening until 11921s when the concentration reaches the target value and then brought down to 80% opening for about 2000s. The steam valve is then again opened to the steady state value of 86%. We also observe that there is some amount of oscillation in operation of the valves every time the valve opening% changes. These rapid changes can be avoided by penalizing the rate constraints on input change as discussed in sections 5.2.1 and 5.2.2.

5.2.5.2 Level of liquid and product concentration inside the evaporator

In this section we see the variation of level inside the evaporation column in this case. Due to 100s prediction horizon the level appears to increase in steps every 100s as depicted below in Figure 5.27a. The level attains a maximum value at about 550s similar to the previous cases. The evaporator level is controlled at 100% level till the desired product concentration is achieved at around 11663s and is brought down to the target range of 64% after the concentration of A in product reaches a value of 0.8. The product concentration also has an identical behavior to the previous sections as depicted in Figure 5.27b. The next section provides a table for comparing the execution times of the various approaches discussed in this section.



(a)



(b)

Figure 5.27a) Level and b) product concentration (single-window, $h_{opt} = 100, h_{sim} = 1$)

5.3 Execution times

In this section we present a brief comparison of the execution times in our work with the results in [1]. Table 5.2 shows the computation times required for arriving at the solution in each of the cases defined in this chapter.

Table 5.2 Computation time comparison

Approach	t_f	CPU time	CPU configuration
Case1: Single-window , $h_{opt} = 1$	15000	3.40 hrs	i5@1.6GHz
Case2: Two-part-window $h_{opt} = 1$	15000	7.35 hrs	i7@3.2GHz
Case3: Two-part-window $h_{opt} = 0.5$	15000	9.74 hrs	i5@1.6GHz
Single-window, $h\_opt = 100$, $h_{sim} = 1$	15000	2.58 hrs	i5@1.6GHz
NLP DAE solver [1] , $h_{opt} = 1$	13937	6.00 hrs	PIV@2.8GHz

It can be seen that the computation time for the single window case using the NLP DAE solver used in [1] is about 6 hours on a P-IV machine with 2.8GHz clock frequency. The time taken for computing the solution in our case is about 3.61 hours on an i5 machine with clock frequency of 1.6GHz. This data is only to present the results of the implementation and clearly this is not any measure of the speed of the algorithm given the differences in the algorithm implementation. However from the comparison of the behavior of process variables in section 5.2.1 we could conclude that the results are comparable to the work in [1]. So we can definitely claim that modeling as an interconnected dynamical system is an alternative approach to mathematically model the evaporation process. The Table 5.2 also shows higher execution time in case 2 and case 3 in comparison to the case 1. Case 2 implemented on a faster system (i7@3.2Ghz) takes about 2.5 times the time required for execution of case 1. Case 2 has two partitions to solve for and hence about 2.5 times rise in CPU time is justified. Case 3 implemented on same system as case 1 takes about 3 times the time taken in case1. Based on the values of $h\_opt = 0.5$ solved for single-windows in case 3 compared to the values of $h_{opt} = 1$ with single-window in case 1, the 3 times rise in CPU time is justified. The case with $h_{opt} = 100$ takes the least amount of time to find a solution. This is justified as less number of steps are required for optimization as $N = t_f/h_{opt}$, but at the same time he time taken for each step is higher in this case as each step needs 100 steps of simulation to be computed.

Figure 5.28 shows computation time for each call made to the $fmincon$ for the single window case. The two sharp peaks correspond to the times when the level and product concentration reach their respective target values. The cost-value changes when these two events occur. It is observed that the average computation time is 0.78s.

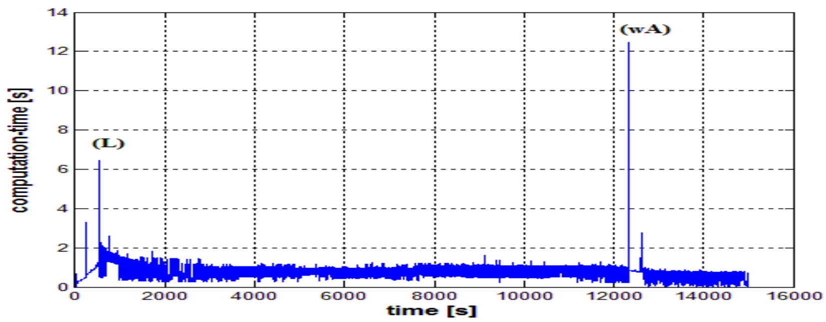


Figure 5.28 Computation time for each call to the fmincon (single-window)

5.4 Mode transition times

Table 5.3 presents the mode transition times, the time when the process shifts into evaporating mode from non-evaporating mode, for all the cases considered in this implementation. We observe that the time spent in the non-evaporating mode during the startup of the evaporation process in all these case is very low. However, we observe from the plots discussed in this chapter that the process variables make a smooth transition between the two modes.

Table 5.3 Mode transition times for test cases

Approach	**Mode transition time**
Case 1: Single-window , $h_{opt} = 1$	56s
Case 2: Two-part-window $h_{opt} = 1$	56s
Case 3: Two-part-window $h_{opt} = 0.5$	55.5s
Two-part-window $h_{opt} = 0.5$ (steam correction)	32s

6. FUTURE WORK

Having implemented a solver for optimal-startup of an evaporation system, in this section we propose future directions to this work. First we analyze the feasibility of real time implementation and online optimization using this approach in section 6.1. Then we discuss the scope for improvement of this work in the last section of this chapter

6.1 Real time implementation

In this section we discuss the feasibility and limitations of implementing an online real time optimizing controller for an evaporation process startup using the computational method developed in this thesis. Clearly the overall startup-time of the system or in other words the time to reach steady state of operation simulated in this case is about 11921s. This is the time when the concentration of component A in product attains the target value of 0.8 as shown in Figure 6.1. The total computation time till 11921s seconds turns out to be 9511s. This indicates that the average computation time for each step is 0.7978s (less than 1s). This shows feasibility of online optimization using this approach.

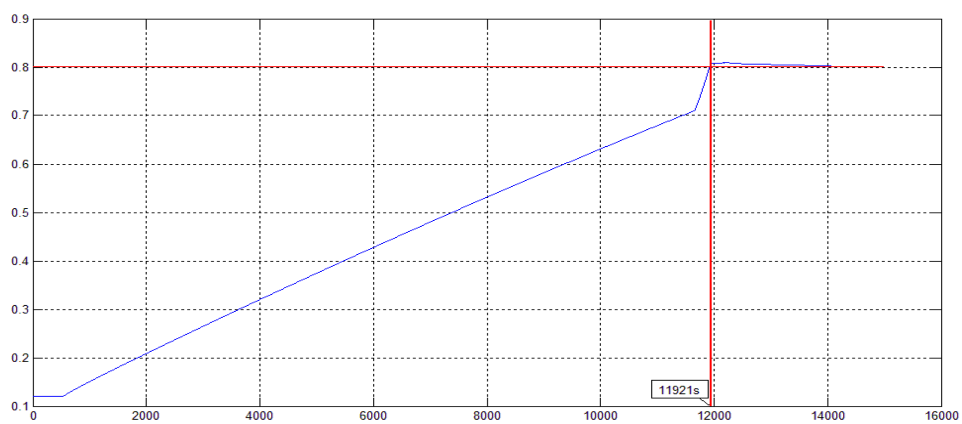


Figure 6.1 Time to steady-state operation

Implementing an online optimizing controller has many hurdles and will need more research. Two of the key issues that can be clearly listed are the losses and delays [33] during operation of real processes. Relevant instances include:

1. Heat-losses in the heat exchanger: These are assumed to be negligible in the modeling assumptions [2]. This can cause the values of the model computed variables to be very different from the real behavior of the process in case of large scale applications.
2. Time delay in process response: In our implementation the high fidelity simulation described in section 4.5 serves as the real process model. The response of a real process to changing inputs might not be instantaneous and the delay might change the optimal input solution computed by the solver.

This is not a comprehensive list of all the issues in real time implementation. However the issues related to losses and delays in the process can be handled by estimating and incorporating the mathematical model of the delays and the losses in the current process model [33]. The model predictive controller might lead to a different profile of the control inputs computed by the solver in that case. However other real time process issues like actuator error and process disturbances like leaks and other transients, are not accounted for in this evaporation model considered in this Thesis. This can be seen as a limitation to this work. However in this case we focus only on optimized startup control of the process and the solver gives useful results regarding the optimal opening sequence for the process valves which is difficult to be determined intuitively.

6.2 Scope for improvement

In this section we present the results obtained by testing the execution of MATLAB code given in Appendix C using MATLAB profiler [32] to analyze the execution times of each function. Table 6.1 presents details of the percentage of time consumed by the $fmincon$, RelaxE and NRE functions. The column self-time indicates the total code time spent inside the function including the time spent in child functions.

Table 6.1 Comparison of function execution times

Function Name	Calls	**Total Time**	Self Time*
mainEvapSolver	1	17622.067 s	5427.252 s
mainEvapSolver>RelaxE	251797	11984.413 s	48.502 s
mainEvapSolver>NRE	1246309	11935.972 s	11935.972 s
fmincon	15000	10967.911 s	12.309 s

It is observed that the NRE function is called the maximum number of times and it consumes the maximum execution time of 11935.972s in the program which turns out to be about 67% (11935.972/17622.067*100) of the total program execution time. NRE is the function which simultaneously computes solution for equations (4.79) and (4.80). This involves matrix operations for solving the system of equations. Hence more research on the efficient computational methods for the specific problem under consideration can lead to huge improvement in the performance of the solver for that problem.

In this work we considered the example of a hybrid non-linear dynamical model of the evaporation process for optimal control using our approach. In the future it will be our endeavor to extend these ideas to real-time optimal control of other non-linearly modelled hybrid systems.

REFERENCES

REFERENCES

[1] Christian Sonntag, Wanjing Su, Olaf Stursberg, Sabastian Engell (2008). *"Optimized start-up control of an industrial-scale evaporation system with hybrid dynamics",* Control Engineering Practice, 16 (2008), 976-990

[2] Christian Sonntag, Olaf Stursberg (2008). *"Optimally controlled start-up of a multi-stage evaporation system",* Technical Report for the HYCON network of excellence, University of Dortmund

[3] Christian Sonntag, Sven Lohmann, Anna Volker, Sabastian Engell (2008). *"Analyzing safety properties of hybrid processing systems: A case study on an industrial evaporator",* Journal of Process Control 18(2008) 885-895

[4] Q.Q. Chai, C.H. Yang, K.L Teo, W.H. Gui (2012). *"Optimal control of an industrial-scale evaporation process: Sodium aluminate solution",* Control Engineering Practice, 20 (2012), 618-628

[5] Elhaq, S. L., Giri, F., & Unbehauen, H. (1999). "*Modeling identification and control of sugar evaporation theoretical design and experimental evaluation*", Control Engineering Practice, 7(8), 931–942.

[6] Kam, K. M., & Tade´, M. O. (1999). "*Nonlinear control of a simulated industrial evaporation system using a feedback linearization technique with a state observer*", Industrial & Engineering Chemistry Research, 38(8), 2995–3006

[7] Loxton, R. C., Teo, K. L., & Rehbock, V. (2008). "*Optimal control problems with multiple characteristic time points in the objective and constraints" Automatica, 44(11), 2923-2929*

[8] Miranda, V., & Simpson, R. (2005). *"Modeling and simulation of an industrial multiple effect evaporator: tomato concentrate",* Journal of Food Engineering, 66(2), 203–210.

[9] Alberto Bemporad, Manfred Morari (1999). *"Control of systems integrating logic, dynamics and constraints"*, Automatica 35 (1999) 407-427

[10] Hans Schuler, 2006, *"Automation in Chemical Industry"*

[11] P. Varaiya, *Reach set computation using optimal control*, In Proceedings of the KIT Workshop on Verification of Hybrid Systems, 377–383, Grenoble, France, 1998.

[12] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. "*Linear Matrix Inequalities in System and Control Theory*". SIAM, Philadelphia, 1994.

[13] C. Mohtadi, S.L. Shah, D.W. Clarke. "*Generalized predictive control of multivariable systems*", System and Control Letters, 9 (1987), p. 295

[14] Kwakernaak, Huibert and Sivan, Raphael (1972). "*Linear Optimal Control Systems. First Edition*." Wiley-Interscience. ISBN 0-471-51110-2.

[15] Meyer, Richard, et al., *"Hybrid model predictive power flow control of a fuel cell-battery vehicle."*, American Control Conference (ACC), 2011. IEEE, 2011.

[16] R.A DeCarlo, Richard Saeks (1981). "*Interconnected Dynamical Systems*" Electrical Engineering and Electronics/10, ISBN 0-8247-6639-3.

[17] L Mockus, G.V. Reklaitis (1997). "*Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization*" , Computers and Chemical Engineering, Volume 21, Issue 10, p. 1147-1156

[18] Kamarizan Kidam, Markku Humme, Mimi H. Hassim (2010). "*Technical analysis of accidents in chemical processes industry and lessons learnt* ", Chemical Engineering Transactions, 19, pp. 451-456

[19] Anselmo Buso and Monica Giomo (2011), "*Mathematical Modeling in chemical Engineering: A Tool to Analyze Complex Systems, Numerical Simulations of Physical and Engineering Processes*" Prof. Jan Awrejcewicz (Ed.),ISBN: 978-953-307-620-1, InTech

[20] Delgass et al. (2013), "*Seventy-Five Years of Chemical Engineering*". Purdue University. Retrieved 13 August 2013

[21] Z Nagy et al. (2004), "*Nonlinear model predictive control: From theory to application*", J. Chin. Inst. Chem. Engrs, 2004, 35, Issue 3, p. 299-315

[22] Z Nagy et al. (2004), "*Nonlinear model predictive control: From theory to application*", J. Chin. Inst. Chem. Engrs, 2004, 35, Issue 3, p. 299-315

[23] Z Nagy et al. (2007), "*Advanced control of a reactive distillation column*", Computer Aided Chemical Engineering 24 (2007): 805.

[24] Prett, David M., Brian L. Ramaker, and Charles R. Cutler. "Dynamic matrix control method." U.S. Patent No. 4,349,869. 14 Sep. 1982.

[25] Kothare, Mayuresh V., Venkataramanan Balakrishnan, and Manfred Morari. "Robust constrained model predictive control using linear matrix inequalities."*Automatica* 32.10 (1996): 1361-1379.

[26] Mitchell, Ian M., Alexandre M. Bayen, and Claire J. Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games."*Automatic Control, IEEE Transactions on* 50.7 (2005): 947-957.

[27] Boltyanskii, Vladimir Grigor'evich, Revaz Valer'yanovich Gamkrelidze, and Lev Semenovich Pontryagin. *The theory of optimal processes. I. The maximum principle*. TRW SPACE TECHNOLOGY LABS LOS ANGELES CALIF, 1960.

[28] Boggs, Paul T., and Jon W. Tolle. "Sequential quadratic programming." *Acta numerica* 4 (1995): 1-51.

[29] Mehrotra, Sanjay. "On the implementation of a primal-dual interior point method." *SIAM Journal on optimization* 2.4 (1992): 575-601.

[30] Agrawal, Om P. "Formulation of Euler–Lagrange equations for fractional variational problems." *Journal of Mathematical Analysis and Applications* 272.1 (2002): 368-379.

[31] Smith, Alice E.; Coit David W. "Penalty functions Handbook of Evolutionary Computation", Section C 5.2. Oxford University Press and Institute of Physics Publishing, 1996.

[32] Mathworks. Inc. (2015). *"MATLAB Documentation"*. Retrieved April 11, 2015 from www.mathworks.com/help/

[33] Nilsson, Johan. "*Real-time control systems with delays*." Diss. Lund institute of Technology, 1998.

[35] Diethelm, Kai, Neville J. Ford, and Alan D. Freed. "A predictor-corrector approach for the numerical solution of fractional differential equations.", *Nonlinear Dynamics* 29.1-4 (2002): 3-22.

[36] Meyer, Richard, Miloš Žefran, and Raymond A. DeCarlo. "A Comparison of the Embedding Method to Multi-Parametric Programming, Mixed-Integer Programming, Gradient-Descent, and Hybrid Minimum Principle Based Methods." *arXiv preprint arXiv:1203.3341* (2012).

APPENDICES

A. ENERGY CONTENT OF THE LIQUID PHASE

In this section we present the calculations for arriving at equation 2.4 using the energy content of vapor phase defined by equation 2.9.

Our goal is to arrive at equation 2.4:

$$\frac{dU}{dt} = \dot{m}_{i,l}.c_{p,l}(w).T_i - \dot{m}_{o,l}.c_{p,l}(w).T + \dot{Q}$$

From the equation 2.9:

$$U = m_l.c_{p,l}(w).T$$

Differentiating both sides of equation 2.9 we have:

$$\dot{U} = \dot{m}_l.c_{p,l}(w).T + m_l.\dot{c}_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T}$$

As the mass fractions of components given by (w) remains constant in the non-evaporating mode due to no evaporation of the components, the second term in this equation is zero.

$$\dot{U} = \dot{m}_l.c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T}$$

Using equation 2.7 the first term can be expanded as:

$$\dot{U} = (\dot{m}_A + \dot{m}_B + \dot{m}_C).c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T}$$

Substituting for $\dot{m}_A, \dot{m}_B$ and $\dot{m}_C$ from equation 2.1 – 2.3 we have:

$$\dot{U} = (\dot{m}_{i,l}.w_{A,i} - \dot{m}_{o,l}.w_A + \dot{m}_{i,l}.w_{B,i} - \dot{m}_{o,l}.w_B + \dot{m}_{i,l}.w_{C,i} - \dot{m}_{o,l}.w_C).c_{p,l}(w).T$$
$$+ m_l.c_{p,l}(w).\dot{T}$$
$$= (\dot{m}_{i,l}.w_{A,i} + \dot{m}_{i,l}.w_{B,i} + \dot{m}_{i,l}.w_{C,i} - \dot{m}_{o,l}.w_A - \dot{m}_{o,l}.w_B - \dot{m}_{o,l}.w_C).c_{p,l}(w).T$$
$$+ m_l.c_{p,l}(w).\dot{T}$$
$$= (\dot{m}_{i,l}.(w_{A,i} + w_{B,i} + w_{C,i}) - \dot{m}_{o,l}.(w_A + w_B + w_C)).c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T}$$

From equation 2.6 we have $(w_{A,i} + w_{B,i} + w_{C,i}) = (w_A + w_B + w_C) = 1$

$$=> \dot{U} = (\dot{m}_{i,l} - \dot{m}_{o,l}).c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T}$$
$$= (\dot{m}_{i,l}.c_{p,l}(w).T - \dot{m}_{o,l}.c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T})$$
$$= (\dot{m}_{i,l}.c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T} - \dot{m}_{o,l}.c_{p,l}(w).T) \qquad \text{(A1.1)}$$

Assuming negligible loss of heat from the system, the heat transferred from the heat exchanger, $\dot{Q}$ is consumed in heating the incoming liquid from a temperature T_i to T and for leading to rise in temperature of the liquid inside the evaporation column. This can be formulated as:

$$\dot{Q} = \dot{m}_{i,l}.c_{p,l}(w).(T - T_i) + m_l.c_{p,l}(w).\dot{T}$$

$$=> \dot{m}_{i,l}.c_{p,l}(w).T + m_l.c_{p,l}(w).\dot{T} = \dot{m}_{i,l}.c_{p,l}(w).T_i + \dot{Q} \qquad \text{(A1.2)}$$

Substituting equation (A1.2) in equation (A1.1) we have:

$\dot{U} = (\dot{m}_{i,l}.c_{p,l}(w).T_i + \dot{Q} - \dot{m}_{o,l}.\, c_{p,l}(w).T)$

Hence we have arrived at equation 2.4 using equation 2.9.

B. THERMODYNAMIC RELATIONSHIPS

In this section we list the thermodynamic relationships governing the mixture of components in liquid and vapor phase as given in reference [2].

Specific heat capacity of mixture in liquid phase $\left(\frac{kJ}{kg.K}\right)$

The specific heat capacity of a mixture of liquids is given as:

$$c_{p,l}(w) = 3.05w_A + 4.315w_B + 3.5w_C$$

where, w_A, w_B, w_C are the mass fractions of respective components A, B and C in liquid.

Enthalpy of evaporation of mixture in vapor phase at $0°C$ $\left(\frac{kJ}{kg.K}\right)$

Enthalpy of evaporation of vapor in the evaporator is given by:

$$h_v(\boldsymbol{\xi}) = 2375.8752\boldsymbol{\xi}_B + 1184.8742\boldsymbol{\xi}_C$$

where, $\boldsymbol{\xi}_B, \boldsymbol{\xi}_C$ are the mass fractions of respective components B and C in vapor.

Specific heat capacity of mixture in vapor phase $\left(\frac{kJ}{kg.K}\right)$

Specific heat of vapor in the evaporator is given by:

$$c_{p,v}(\boldsymbol{\xi}) = 0.0019325\boldsymbol{\xi}_B + 0.0016115\boldsymbol{\xi}_C$$

where, $\boldsymbol{\xi}_B, \boldsymbol{\xi}_C$ are the mass fractions of respective components B and C in vapor.

Energy content of vapor in terms of temperature of evaporator, T

$$f_v(T, \xi) = c_{p,v}(\boldsymbol{\xi})T + h_v(\boldsymbol{\xi})$$

Density of mixture in liquid phase $\left(\frac{kg}{m^3}\right)$

The density of liquid inside the evaporation column is given by:

$$\rho_l(w) = 1046.085\, w_A + 930.42\, w_B + 658.9\, w_C$$

where, w_A, w_B, w_C are the mass fractions of respective components B and C in liquid.

Vapor pressure of component B and C $\left(\boldsymbol{P_B^0}, \boldsymbol{P_C^0}\, [\boldsymbol{bar}]\right)$

Enthalpy of evaporation of vapor in the evaporator is given by:

$$P_B^0(T) = 0.0000050\, T^3 + 0.0048\, T^2 + 1.54\, T - 164.9$$

$$P_C^0(T) = 0.00001038\, T^3 + 0.0098\, T^2 + 3.14\, T - 339.75$$

where, T is the temperature inside the evaporation column

C. MATLAB CODE

In this section we present the code of the algorithm developed for this approach

function mainEvapSolver()

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Non-Evaporating Mode:

xdot(1) = a(1)*C.wAi - a(2)*x(1)/(a(3)+ep)              = f1(x,a)
xdot(2) = a(1)*C.wBi - a(2)*x(2)/(a(3)+ep)              = f2(x,a)
xdot(3) = a(1)*C.wCi - a(2)*x(3)/(a(3)+ep)              = f3(x,a)
xdot(4) = a(1)*C.Cpil*C.Ti - a(2)*x(4)/(a(3)+ep) + a(6) = f4(x,a)

b(1) = ml     = g1(x,a)
b(2) = Phe    = g2(x,a)
b(3) = mdothe = g3(x,a)
b(4) = Q      = g4(x,a)
b(5) = wA     = g5(x,a)
b(6) = wB     = g6(x,a)
b(7) = wC     = g7(x,a)
b(8) = T      = g8(x,a)
b(9) = P      = g9(x,a)
b(10)= L      = g10(x,a)
% 1 2 3 4         5     6 7 8 9 10
[ 0 0 0 0         0     0 0 0 0 0 ] %] = a(1) = mil    = 10*u(1)
[ 0 0 0 0         0     0 0 0 0 0 ] %] = a(2) = mol    = 10*u(2)
[ 0 0 1 0         0     0 0 0 0 0 ] %] = a(3) = mdothe = b(3)
[ 0 0 0 1         0     0 0 0 0 0 ] %] = a(4) = Q      = b(4)
[ 0 0 0 0         1     0 0 0 0 0 ] %] = a(5) = wA     = b(5)             =
[ 0 0 0 0         0     1 0 0 0 0 ] %] = a(6) = wB     = b(6)
[ 0 0 0 0         0     0 1 0 0 0 ] %] = a(7) = wC     = b(7)
[ 0 0 0 0         0     0 0 1 0 0 ] %] = a(8) = T      = b(8)
[ 0 0 0 0         0     0 0 0 0 0 ] %] = a(9) = vv2    = u(4)
[b(9)]  %]
[ 0 0 0 0         0     0 0 0 0 1 ] %] = a(10)= P      = b(10)
[b(10)] %]
[ 1 0 0 0         0     0 0 0 0 0 ] %] = a(11)= ml     = b(1)
[ 0 1 0 0         0     0 0 0 0 0 ] %] = a(12)= Phe    = b(2)
[ 0 0 0 1/(C.K*C.A) 0   0 1 0 0 0 ] %] = a(13)= The    = b(4)/(C.K*C.A) + b(8)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Evaporating mode:

xdot(1,k+1) = a_k(1,k)*C.wAi - a_k(2,k)*a_k(5,k)
= f1(x,a)
xdot(2,k+1) = a_k(1,k)*C.wAi - a_k(2,k)*a_k(6,k) - a_k(3,k)*a_k(7,k)
```

```
= f2(x,a)
xdot(3,k+1) = a_k(1,k)*C.wAi - a_k(2,k)*a_k(7,k) - a_k(3,k)*a_k(7,k)
= f3(x,a)
xdot(4,k+1) = a_k(1,k)*C.Cpil*C.Ti - a_k(2,k)*Cpl*a_k(8,k) - a_k(3,k)*(Cpv*a_k(8,k) +
hve) + a_k(15,k) = f4(x,a)

x0(1) + x0(2) + x0(3) - a0(12)*a0(13)                                    %b(1) = ml
= g1(x,a)
(a0(6)*PB0 + a0(7)*PC0)/(C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC))  %b(2) = rhov
= g2(x,a)
C.V - a0(11)/(1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7))               %b(3) = Vv
= g3(x,a)
(C.Cpvi*C.Tis - C.Cpli*a0(16) + C.hvi)*a0(4)                             %b(4) = Q
= g4(x,a) = (C.Cpvi*C.Tis - Cpli*a(16) + hvi)*a(4)
x0(1)/a0(11)                                                             %b(5) = wA
= g5(x,a)
(x0(2)-a0(9)*a0(12)*a0(13))/a0(11)                                       %b(6) = wB
= g6(x,a)
(x0(3)-a0(10)*a0(12)*a0(13))/a0(11)                                      %b(7) = wC
= g7(x,a)
(x0(4)-(2375.8752*a(9) + 1184.8742*a(10))*a(12)*a(13))/(a0(11)*(3.05*a(5) + 4.315*a(6) +
3.5*a(7)) + a0(12)*a0(13)*(0.0019325*a0(9) + 0.0016115a0(10)))
%b(8) = T       = g8(x,a)
a0(6)*PB0/(a0(6)*PB0 + a0(7)*PC0)                                        %b(9) = EB
= g9(x,a)
a0(7)*PC0/(a0(6)*PB0 + a0(7)*PC0)                                        %b(10)= EC
= g10(x,a)
((a0(11)/(1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7)))-C.VB)/C.vL       %b(11)= L
= g11(x,a)
((a0(6)/C.MB)*PB0 + (a0(7)/C.MC)*PC0)/(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)%b(12)= P
= g12(x,a) = (wA*PB0 + wB*PC0)/((wA/C.MA))+(wB/C.MB))+(wC/C.MC)))
(-0.069 * sqrt(a0(12)) * ((a0(18) - C.Pa)^3 - 36 * (a0(18) - C.Pa)^2 - 69 * (a0(18) -
C.Pa))/((a0(18) - C.Pa)+0.08)) * a(19)    %b(13)= mdot_ov = g13(x,a)
(-0.069 * sqrt(C.rhovi) * ((C.Pi - a0(17))^3 - 36 * (C.Pi - a0(17))^2 - 69 * (C.Pi -
a0(17)))/((C.Pi - a0(17))+0.08)) * a0(14)  %b(14)= mhot_he = g14(x,a)
(C.Pi/C.Ti)*a0(16)                                                       %b(15)= Phe
= g15(x,a)

% L11
%1  2  3    4        5     6  7  8  9  10 11 12 13 14 15
[0  0  0    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(1) = mil      = 10*u(1)
 0  0  0    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(2) = mol      = 10*u(2)
 0  0  0    0        0     0  0  0  0  0  0  0  1  0  0  %] = a(3) = mdot_ov  = b(13)
 0  0  0    0        0     0  0  0  0  0  0  0  0  1  0  %] = a(4) = mdot_he  = b(14)
 0  0  0    0        1     0  0  0  0  0  0  0  0  0  0  %] = a(5) = wA       = b(5)
 0  0  0    0        0     1  0  0  0  0  0  0  0  0  0  %] = a(6) = wB       = b(6)
 0  0  0    0        0     0  1  0  0  0  0  0  0  0  0  %] = a(7) = wC       = b(7)
 0  0  0    0        0     0  0  1  0  0  0  0  0  0  0  %] = a(8) = T        = b(8)
 0  0  0    0        0     0  0  0  1  0  0  0  0  0  0  %] = a(9) = EB       = b(9)
 0  0  0    0        0     0  0  0  0  1  0  0  0  0  0  %] = a(10)= EC       = b(10)
 1  0  0    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(11)= ml       = b(1)
 0  1  0    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(12)= rhov     = b(2)
```

```
 0  0  1    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(13)= Vv        = b(3)
 0  0  0    0        0     0  0  0  0  0  0  0  0  0  0  %] = a(14)= Vv2       = u(4)
 0  0  0    1        0     0  0  0  0  0  0  0  0  0  0  %] = a(15)= Q         = b(4)
 0  0  0 1/(C.K*C.A) 0     0  0  1  0  0  0  0  0  0  0  %] = a(16)= The       =
b(4)/(C.K*C.A) + b(8)
 0  0  0    0        0     0  0  0  0  0  0  0  0  0  1  %] = a(17)= Phe       = b(15)
 0  0  0    0        0     0  0  0  0  0  0  1  0  0  0  %] = a(18)= P         = b(12)
 0  0  0    0        0     0  0  0  0  0  0  0  0  0  0];%] = a(19)= Vv1       = u(3)

%L12
[ 10 0 0 0  %1] = a(1) = mil      = 10*u(1)
  0 10 0 0  %2] = a(2) = mol      = 10*u(2)
  0  0 0 0  %3
  0  0 0 0  %4
  0  0 0 0  %5
  0  0 0 0  %6
  0  0 0 0  %7
  0  0 0 0  %8
  0  0 0 0  %9
  0  0 0 0  %10
  0  0 0 0  %11
  0  0 0 0  %12
  0  0 0 0  %13
  0  0 0 1  %14] = a(14)= Vv2      = u(4)
  0  0 0 0  %15
  0  0 0 0  %16
  0  0 0 0  %17
  0  0 0 0  %18
  0  0 1 0];%19] = a(19)= Vv1      = u(3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hvE = 2375.8752*a(9) + 1184.8742*a(10)
Cpv = 0.0019325*a(9) + 0.0016115*a(10)
Cpl = (3.05*a0(5) + 4.315*a0(6) + 3.5*a0(7))
rho = (1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7))
PB0 = (0.0000050*(a0(8))^3 - 0.0048*(a0(8))^2 + 1.54*(a0(8)) - 164.9)
PC0 = (0.00001038*(a0(8))^3 - 0.0098*(a0(8))^2 + 3.14*(a0(8)) -339.75)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Authors: RITHESH IYER, RA DECARLO, RT MEYER, SCOTT JOHNSON
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}
```

```
clc
clear all

% Time information
t0 =0;                % Starting time
tf = 15;           % Final time
h_opt = 1;            % Prediction window sample time per partition
Npart = 1;            % Number of partitions of prediction window
```

```
Nsim = 2;               % Number of simulation steps in each partition
Nhires = 10;            % Number of high fidelity simulation steps in each partition
h_sim = h_opt/Nsim;     % steplength of simulation

% Constant values
% Inflow parameters
C.wAi = 0.12;          % Concentration of A in feed inflow
C.wBi = 0.85;          % Concentration of B in feed inflow
C.wCi = 0.03;          % Concentration of C in feed inflow
C.Ti = 327;            % Temperature of feed inflow
C.MA = 130;            % Molecular mass of A
C.MB = 20;             % Molecular mass of B
C.MC = 33;             % Molecular mass of C

% Specific heat of feed inflow
C.Cpil = 3.05*C.wAi + 4.315*C.wBi + 3.5*C.wCi;

% Evaporator parameters
C.vL = 0.058;          % Scaling parameter for level
C.VB = 10;             % Volume in tank when L = 0%
C.V = 40;              % Maximum volume of Evaporator
C.R = 0.08314472;      % Universal gas constant
C.Pa = 0.4;            % Ambient atmospheric pressure
C.ep = 0.001;          % Regularization constant
C.Lmax = 100;          % Maximum value of Level

%HX Parameters
C.K = 4;               % Heat transfer coefficient
C.A = 300;             % Surface area of heat exchanger
C.Q0 = 8000;           % Initial calue of heat transferred
C.mdothe0 = 9;         % Initial flow of steam

% Steam parameters
% Reference: http://www2.spiraxsarco.com/us/resources/steam-tables/superheated-steam.asp
% C.Tis  = 440;        % Temperature of incoming fresh steam
% C.Pi   = 5;          % Pressure of incoming fresh steam
% C.Cpvi = 2.31625;    % Specific heat capacity of fresh steam
% C.Cpli = 4.18;       % Specific heat capacity of condensed steam
% C.hvi = 2375.8752;   % Author value of enthalpy of evaporation at 0deg C
% C.rhovi = 3;         % Density of incoming fresh steam

% % Steam parameters
% % According to Authors
C.Tis  = 440;
C.Pi   = 5;
C.Cpvi = 0.0019325;
C.Cpli = 4.315;
C.hvi = 2375.8752;
C.rhovi = 3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Model matrices initialized for mode = 0
%   1 2 3     4       5 6 7 8 9 10
SNE.L11 = [ 0 0 0     0       0 0 0 0 0 0    %] = a(1) = mil    = 10*u(1)
    0 0 0     0       0 0 0 0 0 0    %] = a(2) = mol    = 10*u(2)
    0 0 1     0       0 0 0 0 0 0    %] = a(3) = mdothe = b(3)
    0 0 0     1       0 0 0 0 0 0    %] = a(4) = Q      = b(4)
    0 0 0     0       1 0 0 0 0 0    %] = a(5) = wA     = b(5)
    0 0 0     0       0 1 0 0 0 0    %] = a(6) = wB     = b(6)
    0 0 0     0       0 0 1 0 0 0    %] = a(7) = wC     = b(7)
    0 0 0     0       0 0 0 1 0 0    %] = a(8) = T      = b(8)
    0 0 0     0       0 0 0 0 0 0    %] = a(9) = VV2    = u(4)
    0 0 0     0       0 0 0 0 0 1    %] = a(10)= P      = b(10)
    1 0 0     0       0 0 0 0 0 0    %] = a(11)= ml     = b(1)
    0 1 0     0       0 0 0 0 0 0    %] = a(12)= Phe    = b(2)
    0 0 0 1/(C.K*C.A) 0 0 0 1 0 0 ]; %] = a(13)= The    = b(4)/(C.K*C.A) + b(8)


SNE.L12 = [ 10 0 0
    0 10 0
    0 0 0
    0 0 0
    0 0 0
    0 0 0
    0 0 0
    0 0 0
    0 0 1
    0 0 0
    0 0 0
    0 0 0
    0 0 0];

SNE.L21 = [ 0 0 0 0 0 0 0 0 1 0
    0 0 0 0 0 0 0 0 0 1 ];

SNE.L22 = [ 0 0 0
    0 0 0 ];

% dimensions of system for a and b
SNE.nb = 10;
SNE.na = 13;

% Matrices for householders formula
NE.indexa = 1:SNE.na;
NE.indexb = SNE.na+1:SNE.na+SNE.nb;

NE.eye_J = eye(SNE.na+SNE.nb);
NE.eye_na = eye(SNE.na);

NE.J0 = [zeros(SNE.nb,SNE.na), -eye(SNE.nb); eye(SNE.na), -SNE.L11];
[NE.LJ0 ,NE.UJ0 ] = lu(NE.J0); % block diagonal structure not utilized for speed
NE.LJ0inv = inv(NE.LJ0);
NE.UJ0inv = inv(NE.UJ0);
```

```
NE.AACC = NE.UJ0inv*NE.LJ0inv(:,1:SNE.nb); % Check for optimizing this
NE.AA = NE.AACC(1:SNE.na,:);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Model matrices initialized for mode = 1

% L11
%   1  2  3    4         5      6  7  8  9  10 11 12 13 14 15
SE.L11 = [  0  0  0    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(1) = mil
= 10*u(1)
    0  0  0    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(2) = mol      =
10*u(2)
    0  0  0    0         0      0  0  0  0  0  0  0  1  0  0  %] = a(3) = mdot_ov  = b(13)
    0  0  0    0         0      0  0  0  0  0  0  0  0  1  0  %] = a(4) = mdot_he  = b(14)
    0  0  0    0         1      0  0  0  0  0  0  0  0  0  0  %] = a(5) = wA       = b(5)
    0  0  0    0         0      1  0  0  0  0  0  0  0  0  0  %] = a(6) = wB       = b(6)
    0  0  0    0         0      0  1  0  0  0  0  0  0  0  0  %] = a(7) = wC       = b(7)
    0  0  0    0         0      0  0  1  0  0  0  0  0  0  0  %] = a(8) = T        = b(8)
    0  0  0    0         0      0  0  0  1  0  0  0  0  0  0  %] = a(9) = EB       = b(9)
    0  0  0    0         0      0  0  0  0  1  0  0  0  0  0  %] = a(10)= EC       = b(10)
    1  0  0    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(11)= ml       = b(1)
    0  1  0    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(12)= rhov     = b(2)
    0  0  1    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(13)= Vv       = b(3)
    0  0  0    0         0      0  0  0  0  0  0  0  0  0  0  %] = a(14)= Vv2      = u(4)
    0  0  0    1         0      0  0  0  0  0  0  0  0  0  0  %] = a(15)= Q        = b(4)
    0  0  0 1/(C.K*C.A) 0      0  0  1  0  0  0  0  0  0  0  %] = a(16)= The      =
b(4)/(C.K*C.A) + b(8)
    0  0  0    0         0      0  0  0  0  0  0  0  0  0  1  %] = a(17)= Phe      = b(15)
    0  0  0    0         0      0  0  0  0  0  0  1  0  0  0  %] = a(18)= P        = b(12)
    0  0  0    0         0      0  0  0  0  0  0  0  0  0  0];%] = a(19)= Vv1      = u(3)

SE.L12 = [ 10 0 0 0  %] = a(1) = mil      = 10*u(1)
    0 10 0 0  %] = a(2) = mol      = 10*u(2)
    0  0 0 0  %3
    0  0 0 0  %4
    0  0 0 0  %5
    0  0 0 0  %6
    0  0 0 0  %7
    0  0 0 0  %8
    0  0 0 0  %9
    0  0 0 0  %10
    0  0 0 0  %11
    0  0 0 0  %12
    0  0 0 0  %13
    0  0 0 1  %] = a(14)= Vv2      = u(4)
    0  0 0 0  %15
    0  0 0 0  %16
    0  0 0 0  %17
    0  0 0 0  %18
    0  0 1 0];%] = a(19)= Vv1      = u(3)
```

```
SE.L21 = [0  0  0   0      0    0  0  0  0  0  0  0  0  0  0  %] = y(1) = L       =
b(11)
    0  0  0   0      0    0  0  0  0  0  0  0  0  0  0];%] = y(2) = P       = b(12)

SE.L22 = [ 0 0 0 0
    0 0 0 0];

% dimensions of system for a and b
SE.nb = 15;
SE.na = 19;

% Matrices for householders formula
E.indexa = 1:SE.na;
E.indexb = SE.na+1:SE.na+SE.nb;

E.eye_J = eye(SE.na+SE.nb);
E.eye_na = eye(SE.na);

E.J0 = [zeros(SE.nb,SE.na), -eye(SE.nb); eye(SE.na), -SE.L11];
[E.LJ0 ,E.UJ0 ] = lu(E.J0); % block diagonal structure not utilized for speed
E.LJ0inv = inv(E.LJ0);
E.UJ0inv = inv(E.UJ0);

E.AACC = E.UJ0inv*E.LJ0inv(:,1:SE.nb); % Check for optimizing this
E.AA = E.AACC(1:SE.na,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cost weights
W.sq = 1000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initializing variables for mode = 0
u10 = 1;
u20 = 0;
u30 = 1;

L0 = 1;
ml0 = (L0*C.vL + C.VB)*(1046.085*C.wAi + 930.42*C.wBi + 658.9*C.wCi);
The0 = C.Q0/(C.K*C.A) + C.Ti;
Phe0 = (C.Pi/C.Tis)*(The0);
PB0 = (0.0000050*(C.Ti)^3 - 0.0048*(C.Ti)^2 + 1.54*(C.Ti) - 164.9);
PC0 = (0.00001038*(C.Ti)^3 - 0.0098*(C.Ti)^2 + 3.14*(C.Ti) -339.75);
P0 = ((C.wBi/C.MB)*PB0 + (C.wCi/C.MC)*PC0)/((C.wAi/C.MA)+(C.wBi/C.MB)+(C.wCi/C.MC));
%= P = g9(x,a)


x0 = [ml0*C.wAi                %x1
    ml0*C.wBi                  %x2
    ml0*C.wCi                  %x3
```

```
    ml0*C.Cpil*C.Ti];          %x4

a0 = [10*u10                        %= mil (closed)
    10*u20                          %= mol (closed)
    C.mdothe0                       %= mdothe
    C.Q0                            %= Q
    C.wAi                           %= wA
    C.wBi                           %= wB
    C.wCi                           %= wC
    C.Ti                            %= T
    u30                             %= Vv2
    P0                              %= P
    ml0                             %= ml
    Phe0                            %= Phe
    The0];                          %= The

b0 = [  ml0                         %=  ml
    Phe0                            %=  Phe
    C.mdothe0                       %= mdothe
    C.Q0                            %= Q
    C.wAi                           %= wA
    C.wBi                           %= wB
    C.wCi                           %= wC
    C.Ti                            %= T
    P0                              %=  P
    L0];                            %=  L

% Single window  uguess _ NE
une_guess = [    100
    0
    100
    0  ];

% Scaling
une_guess = une_guess/100;

une_tot(:,1) = une_guess(1:end-1,1);
vne_tot(:,1) = une_guess(end,1);

% Initializing a and b
[ane0w,bne0w] = NRNE(x0,une_guess(1:end-1,1),0,a0,b0,SNE,C,NE);

xne0w = x0;

V.mA(1,:) = xne0w(1,:);
V.mB(1,:) = xne0w(2,:);
V.mC(1,:) = xne0w(3,:);
V.U(1,:) = xne0w(4,:);
V.mil(1,:) = ane0w(1,:);
V.mol(1,:) = ane0w(2,:);
V.mov(1,:) = 0;
V.mhe(1,:) = ane0w(3,:);
```

```
V.Q(1,:) = ane0w(4,:);
V.Phe(1,:) = ane0w(12,:);
V.The(1,:) = ane0w(13,:);
V.wA(1,:) = ane0w(5,:);
V.wB(1,:) = ane0w(6,:);
V.wC(1,:) = ane0w(7,:);
V.T(1,:) = ane0w(8,:);
V.P(1,:) = ane0w(10,:);
V.L(1,:) = bne0w(9,:);
V.EB(1,:) = 0;
V.EC(1,:) = 0;
V.rhov(1,:) = 0;
V.Vv(1,:) = 0;
V.ml(1,:) = ane0w(11,:);
V.V1(1,:) = u10;
V.V2(1,:) = u20;
V.Vv1(1,:) = 0;
V.Vv2(1,:) = u30;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initializing variables for mode = 1
u10 = 1;
u20 = 0;
u30 = 0;
u40 = 1;
L0 = 1;
ml0 = (L0*C.vL + C.VB)*(1046.085*C.wAi + 930.42*C.wBi + 658.9*C.wCi);
The0 = C.Q0/(C.K*C.A) + C.Ti;
Phe0 = (C.Pi/C.Tis)*(The0);
PB0 = (0.0000050*(C.Ti)^3 - 0.0048*(C.Ti)^2 + 1.54*(C.Ti) - 164.9);
PC0 = (0.00001038*(C.Ti)^3 - 0.0098*(C.Ti)^2 + 3.14*(C.Ti) -339.75);
rhov0 = (C.wBi*PB0 + C.wCi*PC0)/(C.R*C.Ti*(C.wAi/C.MA + C.wBi/C.MB + C.wCi/C.MC));
Vv0 = C.V - ml0/(1046.085*C.wAi + 930.42*C.wBi + 658.9*C.wCi);
P0 = ((C.wBi/C.MB)*PB0 + (C.wCi/C.MC)*PC0)/((C.wAi/C.MA)+(C.wBi/C.MB)+(C.wCi/C.MC));
mdotov0 = (-0.069 * sqrt(rhov0) * ((P0 - C.Pa)^3 - 36 * (P0 - C.Pa)^2 - 69 * (P0 -
C.Pa))/((P0 - C.Pa)+0.08)) * u30;
EB0 = C.wBi*PB0/(C.wBi*PB0 + C.wCi*PC0);
EC0 = C.wCi*PC0/(C.wBi*PB0 + C.wCi*PC0);
Cpv0 = (0.0019325*EB0 + 0.0016115*EC0);

% Initializing V
x0 = [ml0*C.wAi                                     %mA
    ml0*C.wBi + rhov0*Vv0*EB0                       %mB
    ml0*C.wCi + rhov0*Vv0*EC0                       %mC
    ml0*C.Cpil*C.Ti + rhov0*Vv0*Cpv0*C.Ti];         %U

a0 = [ 10*u10                   %1 = mil (closed)
    10*u20                      %2 = mol (closed)
    mdotov0                     %3 = mov
    C.mdothe0                   %4 = mhe
    C.wAi                       %5 = wA
```

```
    C.wBi                        %6 = wB
    C.wCi                        %7 = wC
    C.Ti                         %8 = T
    EB0                          %9 = EB
    EC0                          %10= EC
    ml0                          %11= ml
    rhov0                        %12= rhov
    Vv0                          %13= Vv
    u40                          %14= Vv2
    C.Q0                         %15= Q
    The0                         %16= The
    Phe0                         %17= Phe
    P0                           %18= P
    u30                          %19= Vv1
    ];

b0 = [  ml0                         %1= ml
    rhov0                           %2= rhov
    Vv0                             %3= Vv
    C.Q0                            %4= Q
    C.wAi                           %5= wA
    C.wBi                           %6= wB
    C.wCi                           %7= wC
    C.Ti                            %8= T
    EB0                             %9= EB
    EC0                             %10= EC
    L0                              %11= L
    P0                              %12= P
    mdotov0                         %13= mov
    C.mdothe0                       %14= mhe
    Phe0                            %15= Phe
    ];

% Single window uguess _ E
ue_guess = [    100
    0
    0
    100
    0   ];
% Scaling
ue_guess = ue_guess/100;
ue_tot(:,1) = ue_guess(1:end-1,1);
ve_tot(:,1) = 0*ue_guess(end,1);

% Initializing a and b
[ae0w,be0w] = NRE(x0,ue_guess(1:end-1,1),0,a0,b0,SE,C,E);

xe0w = x0;

% V.mA(1,:) = xe0w(1,:);
% V.mB(1,:) = xe0w(2,:);
% V.mC(1,:) = xe0w(3,:);
```

```
% V.U(1,:) = xe0w(4,:);
% V.mil(1,:) = ae0w(1,:);
% V.mol(1,:) = ae0w(2,:);
V.mov(1,:) = ae0w(3,:);
% V.mhe(1,:) = ae0w(4,:);
% V.Q(1,:) = ae0w(15,:);
% V.Phe(1,:) = ae0w(17,:);
% V.The(1,:) = ae0w(16,:);
% V.wA(1,:) = ae0w(5,:);
% V.wB(1,:) = ae0w(6,:);
% V.wC(1,:) = ae0w(7,:);
% V.T(1,:) = ae0w(8,:);
% V.P(1,:) = ae0w(18,:);
% V.L(1,:) = be0w(11,:);
V.EB(1,:) = ae0w(9,:);
V.EC(1,:) = ae0w(10,:);
V.rhov(1,:) = ae0w(12,:);
V.Vv(1,:) = ae0w(13,:);
% V.ml(1,:) = ae0w(11,:);
% V.V1(1,:) = u10;
% V.V2(1,:) = u20;
V.Vv1(1,:) = u30;
% V.Vv2(1,:) = u40;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fmincon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining bounds for mode = 0
lbne = [0*ones(1,Npart);0*ones(1,Npart);80*ones(1,Npart);zeros(1,Npart)];
ubne = [100*ones(1,Npart);11.5*ones(1,Npart);100*ones(1,Npart);zeros(1,Npart)];

% Scaling
lbne = lbne/100;
ubne = ubne/100;

% Defining bounds for mode = 1
lbe = [0*ones(1,Npart);0*ones(1,Npart);0*ones(1,Npart);80*ones(1,Npart);zeros(1,Npart)];
ube =
[100*ones(1,Npart);0*ones(1,Npart);100*ones(1,Npart);100*ones(1,Npart);zeros(1,Npart)];

% Scaling
lbe = lbe/100;
ube = ube/100;

% bounds for variables: L, P, T, The
xb = [0 100;0 5;300 470;300 470];

%optimset
options = optimset('Display','off','TolFun',1e-3,'TolX',1e-3,'Algorithm','active-set');
exitflag = 5;
output = ' ';
N = (tf - t0)/h_opt;
```

```
stat_looptime = zeros(1,tf/h_opt+1);
stat_comptime = zeros(1,tf/h_opt+1);
mode = zeros(1,tf/h_opt+1);
mode(1) = 0;
warning('off','all');
swe2ne = 1;
swne2e = 1;
```

```
for i = 1:N
    clc
%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     % Display iteration details:
%     disp(['tf = ',num2str(tf),', tw = ',num2str(Npart)]);
%     disp(['Running MPC with h_hopt = ',num2str(h_opt),'s, h_sim = ',...
%         num2str(h_sim),'s, h_hires = ',num2str(h_opt/Nhires),'s']);
%     disp(['step: ',num2str(i)]);
%     disp(['exitflag: ',num2str(exitflag)]);
%     if (i>1)
%         disp(['step time: ',num2str(stat_looptime(i-1)),'s']);
%         disp(['comp time: ',num2str(stat_comptime(i-1)),'s']);
%     end
%     disp(['Time elapsed: ',num2str(sum(stat_looptime)),'s']);
%
%     output
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    [~,x_refsim] = x_reference(h_opt*(i-1), h_opt*(i+Npart-1), h_sim, V);

    if mode(i) == 0
        swne2e = 1;
        % Non Evaporative Stage
        tic
        % Call FMINCON
        % Starting with u_guess and calculating optimal u
        [uv_opt,~,exitflag,output] = fmincon(@(uv)
costFuncNE(uv,xne0w,ane0w,bne0w,x_refsim,xb,Nsim,Npart,h_opt*(i-1),h_opt*(i+Npart-
1),h_sim,h_opt,W,SNE,C,NE)...
            ,une_guess,[],[],[],[],lbne,ubne,[],options);
        stat_comptime(i) = toc;
        % Store u,v values
        une_tot(:,i+1) = uv_opt(1:end-1,1);
        vne_tot(:,i+1) = uv_opt(end,1);

        % Call hires relaxation for first partition
        [x_kw1, a_kw, b_kw] =
RelaxNE(xne0w,ane0w,bne0w,une_tot(:,i+1),0*une_tot(:,i+1),vne_tot(:,i+1),Nhires,1,h_opt*(
i-1),h_opt*(i),SNE,C,NE);

        % Update x0w and u_guess

        xne0w = x_kw1(:,end);
        ane0w = a_kw(:,end);
```

```
        bne0w = b_kw(:,end);

        une_guess = [uv_opt(:,2:end),uv_opt(:,end)];

        V.mA(i+1) = x_kw1(1,end);
        V.mB(i+1) = x_kw1(2,end);
        V.mC(i+1) = x_kw1(3,end);
        V.U(i+1) = x_kw1(4,end);
        V.mil(i+1) = a_kw(1,end);
        V.mol(i+1) = a_kw(2,end);
        V.mov(i+1) = 0;
        V.mhe(i+1) = a_kw(3,end);
        V.Q(i+1) = a_kw(4,end);
        V.Phe(i+1) = a_kw(12,end);
        V.The(i+1) = a_kw(13,end);
        V.wA(i+1) = a_kw(5,end);
        V.wB(i+1) = a_kw(6,end);
        V.wC(i+1) = a_kw(7,end);
        V.T(i+1) = a_kw(8,end);
        V.P(i+1) = a_kw(10,end);
        V.L(i+1) = b_kw(9,end);
        V.EB(i+1) = 0;
        V.EC(i+1) = 0;
        V.rhov(i+1) = 0;
        V.Vv(i+1) = 0;
        V.ml(i+1) = a_kw(11,end);
        V.V1(i+1) = une_tot(1,i+1);
        V.V2(i+1) = une_tot(2,i+1);
        V.Vv1(i+1) = 0;
        V.Vv2(i+1) = une_tot(3,i+1);

    end
    if mode(i) == 1
        swe2ne = 1;
        % Evaporative Stage
        tic
        % Call FMINCON
        % Starting with u_guess and calculating optimal u
        [uv_opt,~,exitflag,output] = fmincon(@(uv)
costFuncE(uv,xe0w,ae0w,be0w,x_refsim,xb,Nsim,Npart,h_opt*(i-1),h_opt*(i+Npart-
1),h_sim,h_opt,W,SE,C,E)...
            ,ue_guess,[],[],[],[],lbe,ube,[],options);
        stat_comptime(i) = toc;
        % Store u,v values
        ue_tot(:,i+1) = uv_opt(1:end-1,1);
        ve_tot(:,i+1) = uv_opt(end,1);

        % Call hires relaxation for first partition
        [x_kw1, a_kw, b_kw] =
RelaxE(xe0w,ae0w,be0w,ue_tot(:,i+1),0*ue_tot(:,i+1),ve_tot(:,i+1),Nhires,1,h_opt*(i-
1),h_opt*(i),SE,C,E);
```

```
        % Update x0w and u_guess

        xe0w = x_kw1(:,end);
        ae0w = a_kw(:,end);
        be0w = b_kw(:,end);

        ue_guess = [uv_opt(:,2:end),uv_opt(:,end)];

        V.mA(i+1) = x_kw1(1,end);
        V.mB(i+1) = x_kw1(2,end);
        V.mC(i+1) = x_kw1(3,end);
        V.U(i+1) = x_kw1(4,end);
        V.mil(i+1) = a_kw(1,end);
        V.mol(i+1) = a_kw(2,end);
        V.mov(i+1) = a_kw(3,end);
        V.mhe(i+1) = a_kw(4,end);
        V.Q(i+1) = a_kw(15,end);
        V.Phe(i+1) = a_kw(17,end);
        V.The(i+1) = a_kw(16,end);
        V.wA(i+1) = a_kw(5,end);
        V.wB(i+1) = a_kw(6,end);
        V.wC(i+1) = a_kw(7,end);
        V.T(i+1) = a_kw(8,end);
        V.P(i+1) = a_kw(18,end);
        V.L(i+1) = b_kw(11,end);
        V.EB(i+1) = a_kw(9,end);
        V.EC(i+1) = a_kw(10,end);
        V.rhov(i+1) = a_kw(12,end);
        V.Vv(i+1) = a_kw(13,end);
        V.ml(i+1) = a_kw(11,end);
        V.V1(i+1) = ue_tot(1,i+1);
        V.V2(i+1) = ue_tot(2,i+1);
        V.Vv1(i+1) = ue_tot(3,i+1);
        V.Vv2(i+1) = ue_tot(4,i+1);

    end
    disp(['|wA=',num2str(V.wA(i+1)),' | L=',num2str(V.L(i+1)),' | P=',num2str(V.P(i+1)),'
| mode=',num2str(mode(i)),' |']);
    % Mode transition conditions
    if V.P(i+1) >=0.4
        mode(i+1) = 1;
        % State resets
        if (swne2e == 1)
            PB0 = (0.0000050*(V.T(i+1))^3 - 0.0048*(V.T(i+1))^2 + 1.54*(V.T(i+1)) -
164.9);
            PC0 = (0.00001038*(V.T(i+1))^3 - 0.0098*(V.T(i+1))^2 + 3.14*(V.T(i+1)) -
339.75);
            EB = V.wB(i+1)*PB0/(V.wB(i+1)*PB0 + V.wC(i+1)*PC0);
            EC = 1-EB;
            Vv = C.V - V.ml(i+1)/(1046.085*V.wA(i+1) + 930.42*V.wB(i+1) +
658.9*V.wC(i+1));
            rhov = (V.wB(i+1)*PB0 + V.wC(i+1)*PC0)/(C.R*V.T(i+1)*(V.wA(i+1)/C.MA +
```

```
V.wB(i+1)/C.MB + V.wC(i+1)/C.MC));
            xe0w = [V.mA(i+1);V.mB(i+1);V.mC(i+1);V.U(i+1)];
            ae0w =
[V.mil(i+1);V.mol(i+1);V.mov(i+1);V.mhe(i+1);V.wA(i+1);V.wB(i+1);V.wC(i+1);V.T(i+1);EB;EC
;V.ml(i+1);rhov;Vv;V.Vv2(i+1);V.Q(i+1);V.The(i+1);V.Phe(i+1);V.P(i+1);V.Vv1(i+1)];
            be0w =
[V.ml(i+1);rhov;Vv;V.Q(i+1);V.wA(i+1);V.wB(i+1);V.wC(i+1);V.T(i+1);EB;EC;V.ml(i+1);V.P(i+
1);V.mov(i+1);V.mhe(i+1);V.Phe(i+1)];
            swne2e = 2;
        end
        if V.wA(i+1)>0.8
            ube =
[100*ones(1,Npart);11.5*ones(1,Npart);100*ones(1,Npart);100*ones(1,Npart);zeros(1,Npart)]
;
            % Scaling
            ube = ube/100;
        end
    else
        mode(i+1) = 0;
        % State resets
        if (swe2ne == 1)
            xne0w = [V.mA(i+1);V.mB(i+1);V.mC(i+1);V.U(i+1)];
            ane0w =
[V.mil(i+1);V.mol(i+1);V.mhe(i+1);V.Q(i+1);V.wA(i+1);V.wB(i+1);V.wC(i+1);V.T(i+1);V.Vv2(i
+1);V.P(i+1);V.ml(i+1);V.Phe(i+1);V.The(i+1)];
            bne0w =
[V.ml(i+1);V.Phe(i+1);V.mhe(i+1);V.Q(i+1);V.wA(i+1);V.wB(i+1);V.wC(i+1);V.T(i+1);V.P(i+1)
;V.L(i+1)];
            swe2ne = 2;
        end
    end
    stat_looptime(i) = toc;
    save('./mainEvapSolverVars.mat','V','stat_looptime','mode','stat_comptime');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Display results

%printV(tf,t0,h_opt,V,mode);

disp('mainEvapSolver completed');

end

function [cost] =
costFuncNE(uv,x0,a0,b0,x_refsim,xb,Nsim,Npart,t0,tf,h_sim,h_opt,W,S,C,NE)

[x_k, a_k, b_k] = RelaxNE(x0,a0,b0,uv(1:end-1,:),0*uv(1:end-
1,:),uv(end,:),Nsim,Npart,t0,tf,S,C,NE);

% Computing J
if (((x_refsim(1,:))-a_k(5,end))>0.09)
    x_err = [4*W.sq*(x_refsim(1,:) - a_k(5,:)).^2;   % wA error
```

```
        W.sq*(x_refsim(2,:) - b_k(9,:)).^2            ]; % L  error
else
    x_err = [W.sq*(x_refsim(1,:) - a_k(5,:)).^2;   % WA error
        4*W.sq*(x_refsim(2,:) - b_k(9,:)).^2            ]; % L  error
end
% Computing state violation and input cost
costx = 0; costu = 0; costv = 0;
% % L violation
if(sum(b_k(10,:)<xb(1,1))>=1 || sum(b_k(10,:)>xb(1,2))>=1)
    costv = costv + W.sq*sum((xb(1,1) - b_k(10,b_k(10,:)<xb(1,1))).^2) +
W.sq*sum((b_k(10,b_k(10,:)>xb(1,2)) - xb(1,2)).^2);
end
% P violation
if(sum(b_k(9,:)<xb(2,1))>=1 || sum(b_k(9,:)>xb(2,2))>=1)
    costv = costv + W.sq*sum((xb(2,1) - b_k(9,b_k(9,:)<xb(2,1))).^2) +
W.sq*sum((b_k(9,b_k(9,:)>xb(2,2)) - xb(2,2)).^2);
end
% T violation
if(sum(b_k(8,:)<xb(3,1))>=1 || sum(b_k(8,:)>xb(3,2))>=1)
    costv = costv + W.sq*sum((xb(3,1) - b_k(8,b_k(8,:)<xb(3,1))).^2) +
W.sq*sum((b_k(8,b_k(8,:)>xb(3,2)) - xb(3,2)).^2);
end
% The violation
if(sum(a_k(13,:)<xb(4,1))>=1 || sum(a_k(13,:)>xb(4,2))>=1)
    costv = costv + W.sq*sum((xb(4,1) - a_k(13,a_k(13,:)<xb(4,1))).^2) +
W.sq*sum((a_k(13,a_k(13,:)>xb(4,2)) - xb(4,2)).^2);
end
% Total cost
cost = h_sim*costx + h_opt*costu + costv + sum(sum(x_err));

end
```

```
function [cost] = costFuncE(uv,x0,a0,b0,x_refsim,xb,Nsim,Npart,t0,tf,h_sim,h_opt,W,S,C,E)

[x_k, a_k, b_k] = RelaxE(x0,a0,b0,uv(1:end-1,:),0*uv(1:end-
1,:),uv(end,:),Nsim,Npart,t0,tf,S,C,E);

% Computing J

if (((x_refsim(1,:))-a_k(5,end))>0.09)
    x_err = [4*W.sq*(x_refsim(1,:) - a_k(5,:)).^2;   % WA error
        W.sq*(x_refsim(2,:) - b_k(11,:)).^2            ]; % L  error
else
    x_err = [W.sq*(x_refsim(1,:) - a_k(5,:)).^2;   % WA error
        4*W.sq*(x_refsim(2,:) - b_k(11,:)).^2          ]; % L  error
end
% Computing state violation and input cost
costx = 0; costu = 0; costv = 0;

% % L violation
if(sum(b_k(11,:)<xb(1,1))>=1 || sum(b_k(11,:)>xb(1,2))>=1)
    costv = costv + W.sq*sum((xb(1,1) - b_k(11,b_k(11,:)<xb(1,1))).^2) +
```

```
W.sq*sum((b_k(11,b_k(11,:)>xb(1,2)) - xb(1,2)).^2);
end
% P violation
if(sum(b_k(12,:)<xb(2,1))>=1 || sum(b_k(12,:)>xb(2,2))>=1)
    costv = costv + W.sq*sum((xb(2,1) - b_k(12,b_k(12,:)<xb(2,1))).^2) +
W.sq*sum((b_k(12,b_k(12,:)>xb(2,2)) - xb(2,2)).^2);
end
% T violation
if(sum(b_k(8,:)<xb(3,1))>=1 || sum(b_k(8,:)>xb(3,2))>=1)
    costv = costv + W.sq*sum((xb(3,1) - b_k(8,b_k(8,:)<xb(3,1))).^2) +
W.sq*sum((b_k(8,b_k(8,:)>xb(3,2)) - xb(3,2)).^2);
end
% The violation
if(sum(a_k(16,:)<xb(4,1))>=1 || sum(a_k(16,:)>xb(4,2))>=1)
    costv = costv + W.sq*sum((xb(4,1) - a_k(16,a_k(16,:)<xb(4,1))).^2) +
W.sq*sum((a_k(16,a_k(16,:)>xb(4,2)) - xb(4,2)).^2);
end
% Total cost
cost = h_sim*costx + h_opt*costu + costv + sum(sum(x_err));

end
```

```
function [x_k,a_k,b_k] = RelaxNE(x0,a0,b0,u1,u2,v,Nsim,Npart,t0,tf,S,C,NE)

hx = (tf-t0)/(Npart*Nsim);
x_k = zeros(numel(x0),Npart*Nsim+1);
a_k = zeros(numel(a0),Npart*Nsim+1);
b_k = zeros(numel(b0),Npart*Nsim+1);
x_k(:,1)=x0;
a_k(:,1)=a0;
b_k(:,1)=b0;

for k = 1:Npart*Nsim
    % a1 b1 denotes mode 1, a2 b2 denotes mode 2
    index = ceil(k/Npart);
    % Single window simulation
    if (Npart ==1)
        index =1;
    end
    % With x
    %         xdot_k = [   a_k(1,k)*C.wAi - a_k(2,k)*x_k(1,k)/(a_k(11,k)+C.ep);
%= f1(x,a)
    %             a_k(1,k)*C.wBi - a_k(2,k)*x_k(2,k)/(a_k(11,k)+C.ep);
%= f2(x,a)
    %             a_k(1,k)*C.wCi - a_k(2,k)*x_k(3,k)/(a_k(11,k)+C.ep);
%= f3(x,a)
    %             a_k(1,k)*C.Cpil*C.Ti - a_k(2,k)*x_k(4,k)/(a_k(11,k)+C.ep) + a_k(4,k)];
%= f4(x,a)
    %
    % Without x
    xdot_k = [   a_k(1,k)*C.wAi - a_k(2,k)*a_k(5,k);                  %= f1(x,a)
        a_k(1,k)*C.wBi - a_k(2,k)*a_k(6,k);                  %= f2(x,a)
```

```
        a_k(1,k)*C.wCi - a_k(2,k)*a_k(7,k);                  %= f3(x,a)
        a_k(1,k)*C.Cpil*C.Ti - a_k(2,k)*(3.05*a_k(5,k) + 4.315*a_k(6,k) +
3.5*a_k(7,k))*a_k(8,k) + a_k(4,k)]; %= f4(x,a)

    % Forward Euler for xkp1
    x_k(:,k+1) = x_k(:,k)+hx*xdot_k;

    % Coll. Corrector
    xmp = (x_k(:,k+1)+x_k(:,k))/2;
    [a_mp,~] = NRNE(xmp,u1(:,index),0,a_k(:,k),b_k(:,k),S,C,NE);
    % With x
    %         xdot_mp =    [a_mp(1,:)*C.wAi - a_mp(2,:)*xmp(1,:)/(a_mp(11,:)+C.ep);
%= f1(x,a)
    %             a_mp(1,:)*C.wBi - a_mp(2,:)*xmp(2,:)/(a_mp(11,:)+C.ep);
%= f2(x,a)
    %             a_mp(1,:)*C.wCi - a_mp(2,:)*xmp(3,:)/(a_mp(11,:)+C.ep);
%= f3(x,a)
    %             a_mp(1,:)*C.Cpil*C.Ti - a_mp(2,:)*xmp(4,:)/(a_mp(11,:)+C.ep) +
a_mp(4,:)]; %= f4(x,a)
    %
    % Without x
    xdot_mp =    [a_mp(1,:)*C.wAi - a_mp(2,:)*a_mp(5,:);          %= f1(x,a)
        a_mp(1,:)*C.wBi - a_mp(2,:)*a_mp(6,:);                    %= f2(x,a)
        a_mp(1,:)*C.wCi - a_mp(2,:)*a_mp(7,:);                    %= f3(x,a)
        a_mp(1,:)*C.Cpil*C.Ti - a_mp(2,:)*(3.05*a_mp(5,:) + 4.315*a_mp(6,:) +
3.5*a_mp(7,:))*a_mp(8,:) + a_mp(4,:)]; %= f4(x,a)

    x_k(:,k+1) = x_k(:,k)+hx*xdot_mp;
    [a_k(:,k+1),b_k(:,k+1)] = NRNE(x_k(:,k+1),u1(:,index),0,a_k(:,k),b_k(:,k),S,C,NE);
end
end
```

```
function [x_k,a_k,b_k] = RelaxE(x0,a0,b0,u1,u2,v,Nsim,Npart,t0,tf,S,C,E)

hx = (tf-t0)/(Npart*Nsim);

x_k = zeros(numel(x0),Npart*Nsim+1);
a_k = zeros(numel(a0),Npart*Nsim+1);
b_k = zeros(numel(b0),Npart*Nsim+1);
x_k(:,1)=x0;
a_k(:,1)=a0;
b_k(:,1)=b0;

for k = 1:Npart*Nsim
    % a1 b1 denotes mode 1, a2 b2 denotes mode 2
    index = ceil(k/Npart);
    % Single window simulation
    if (Npart ==1)
        index =1;
    end
    % Without x
    xdot_k = [ a_k(1,k)*C.wAi - a_k(2,k)*a_k(5,k);
```

```
%= f1(x,a)
        a_k(1,k)*C.wBi - a_k(2,k)*a_k(6,k) - a_k(3,k)*a_k(9,k);
%= f2(x,a)
        a_k(1,k)*C.wCi - a_k(2,k)*a_k(7,k) - a_k(3,k)*a_k(10,k);
%= f3(x,a)
        a_k(1,k)*C.Cpil*C.Ti - a_k(2,k)*(3.05*a_k(5,k) + 4.315*a_k(6,k) +
3.5*a_k(7,k))*a_k(8,k) - a_k(3,k)*((0.0019325*a_k(9,k) + 0.0016115*a_k(10,k))*a_k(8,k)+
(2375.8752*a_k(9,k) + 1184.8742*a_k(10,k))) + a_k(15,k)]; %= f4(x,a)

    % Forward Euler for xkp1
    x_k(:,k+1) = x_k(:,k)+hx*xdot_k;

    % Coll. Corrector
    % Without x
    xmp = (x_k(:,k+1)+x_k(:,k))/2;
    [a_mp,~] = NRE(xmp,u1(:,index),0,a_k(:,k),b_k(:,k),S,C,E);
    xdot_mp =    [ a_mp(1,:)*C.wAi - a_mp(2,:)*a_mp(5,:);
%= f1(x,a)
        a_mp(1,:)*C.wBi - a_mp(2,:)*a_mp(6,:) - a_mp(3,:)*a_mp(9,:);
%= f2(x,a)
        a_mp(1,:)*C.wCi - a_mp(2,:)*a_mp(7,:) - a_mp(3,:)*a_mp(10,:);
%= f3(x,a)
        a_mp(1,:)*C.Cpil*C.Ti - a_mp(2,:)*(3.05*a_mp(5,:) + 4.315*a_mp(6,:) +
3.5*a_mp(7,:))*a_mp(8,:) - a_mp(3,:)*((0.0019325*a_mp(9,:) +
0.0016115*a_mp(10,:))*a_mp(8,:)+ (2375.8752*a_mp(9,:) + 1184.8742*a_mp(10,:))) +
a_mp(15,:)]; %= f4(x,a)

    x_k(:,k+1) = x_k(:,k)+hx*xdot_mp;
    [a_k(:,k+1),b_k(:,k+1)] = NRE(x_k(:,k+1),u1(:,index),0,a_k(:,k),b_k(:,k),S,C,E);
    %     end
end
end
```

```
function [a,b] = NRNE(x0,u,v,a0,b0,S,C,NE)

% Newton Raphson Algorithm for solving for a,b
%{
    Initial guesses are: a0 , b0
    Variables Passed from Relaxation Algo: x0, u, v

    v takes values in {0,1}
    Options:
    Tolerances: Xtol (default (1e-2)), Ftol (default (1e-2))
%}
% Calling the Gfun to obtain G and DG
L11 = S.L11;
L12 = S.L12;

% Find Value of G
PB0 = (0.0000050*(a0(8))^3 - 0.0048*(a0(8))^2 + 1.54*(a0(8)) - 164.9);
PC0 = (0.00001038*(a0(8))^3 - 0.0098*(a0(8))^2 + 3.14*(a0(8)) -339.75);
```

```
% G
G = [   x0(1) + x0(2) + x0(3)                                                             %=
b(1) = ml     = g1(x,a)
    (C.Pi/C.Tis)*a0(13);                                                                  %=
b(2) = Phe    = g2(x,a)
    (-0.069 * sqrt(C.rhovi) * ((C.Pi - a0(12))^3 - 36 * (C.Pi - a0(12))^2 - 69 * (C.Pi -
a0(12)))/((C.Pi - a0(12))+0.08)) * a0(9); %= b(3) = mdothe = g3(x,a)
    (C.Cpvi*C.Tis - C.Cpli*a0(13) + C.hvi)*a0(3);                                         %=
b(4) = Q      = g4(x,a)
    x0(1)/(a0(11) + C.ep);                                                                %=
b(5) = wA    = g5(x,a)
    x0(2)/(a0(11) + C.ep);                                                                %=
b(6) = wB    = g6(x,a)
    x0(3)/(a0(11) + C.ep);                                                                %=
b(7) = wC    = g7(x,a)
    x0(4)/((a0(11) + C.ep)*(3.05*a0(5) + 4.315*a0(6) + 3.5*a0(7)));                      %=
b(8) = T     = g8(x,a)
    ((a0(11)/(1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7))) - C.VB)/C.vL;               %=
b(9) = L     = g9(x,a)
    ((a0(6)/C.MB)*PB0 + (a0(7)/C.MC)*PC0)/((a0(5)/C.MA)+(a0(6)/C.MB)+(a0(7)/C.MC))]; %=
b(10)= P     = g10(x,a)

% DG
dg2_a13 = (C.Pi/C.Tis);
dg3_a9 = (-0.069 * sqrt(C.rhovi) * ((C.Pi - a0(12))^3 - 36 * (C.Pi - a0(12))^2 - 69 *
(C.Pi - a0(12)))/((C.Pi - a0(12))+0.08));
dg3_a12 = (69*C.rhovi^(1/2)*a0(9)*(69*C.Pi - 69*a0(12) + 36*(C.Pi - a0(12))^2 - (C.Pi -
a0(12))^3))/(1000*(C.Pi - a0(12) + 2/25)^2) - (69*C.rhovi^(1/2)*a0(9)*(72*C.Pi -
72*a0(12) - 3*(C.Pi - a0(12))^2 + 69))/(1000*(C.Pi - a0(12) + 2/25));
dg4_a3 = (C.Cpvi*C.Tis - C.Cpli*a0(13) + C.hvi);
dg4_a13 = -C.Cpli*a0(3);
dg5_a11 = -x0(1)/((a0(11) + C.ep)^2);
dg6_a11 = -x0(2)/((a0(11) + C.ep)^2);
dg7_a11 = -x0(3)/((a0(11) + C.ep)^2);
dg8_a5 = -(61*x0(4))/(20*(C.ep + a0(11))*((61*a0(5))/20 + (863*a0(6))/200 +
(7*a0(7))/2)^2);
dg8_a6 = -(863*x0(4))/(200*(C.ep + a0(11))*((61*a0(5))/20 + (863*a0(6))/200 +
(7*a0(7))/2)^2);
dg8_a7 = -(7*x0(4))/(2*(C.ep + a0(11))*((61*a0(5))/20 + (863*a0(6))/200 +
(7*a0(7))/2)^2);
dg8_a11 = -x0(4)/(((a0(11) + C.ep)^2)*(3.05*a0(5) + 4.315*a0(6) + 3.5*a0(7)));
dg9_a5 = -
(1150182621142057*a0(11))/(1099511627776*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg9_a6 = -(46521*a0(11))/(50*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg9_a7 = -(6589*a0(11))/(10*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg9_a11 = 1/(C.vL*((1150182621142057*a0(5))/1099511627776 + (46521*a0(6))/50 +
(6589*a0(7))/10));
dg10_a5 = -((PB0*a0(6))/C.MB + (PC0*a0(7))/C.MC)/(C.MA*(a0(5)/C.MA + a0(6)/C.MB +
a0(7)/C.MC)^2);
```

```
dg10_a6 = PB0/(C.MB*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - ((PB0*a0(6))/C.MB +
(PC0*a0(7))/C.MC)/(C.MB*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg10_a7 = PC0/(C.MC*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - ((PB0*a0(6))/C.MB +
(PC0*a0(7))/C.MC)/(C.MC*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg10_a8 = ((a0(6)*((17708874310761171*a0(8)^2)/1180591620717411303424 - (6*a0(8))/625 +
77/50))/C.MB + (a0(7)*((18381811534570095*a0(8)^2)/590295810358705651712 -
(49*a0(8))/2500 + 157/50))/C.MC)/(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC);

%   1 2     3     4      5       6       7        8       9      10        11     12      13
DG = [  0 0     0     0      0       0       0        0       0       0         0      0       0
%1
    0 0     0     0      0       0       0        0       0       0         0      0    dg2_a13
%2
    0 0     0     0      0       0       0        0     dg3_a9    0         0   dg3_a12    0
%3
    0 0  dg4_a3   0      0       0       0        0       0       0         0      0    dg4_a13
%4
    0 0     0     0      0       0       0        0       0       0      dg5_a11   0       0
%5
    0 0     0     0      0       0       0        0       0       0      dg6_a11   0       0
%6
    0 0     0     0      0       0       0        0       0       0      dg7_a11   0       0
%7
    0 0     0     0   dg8_a5  dg8_a6  dg8_a7      0       0       0      dg8_a11   0       0
%8
    0 0     0     0   dg9_a5  dg9_a6  dg9_a7      0       0       0      dg9_a11   0       0
%9
    0 0     0     0 dg10_a5 dg10_a6 dg10_a7 dg10_a8   0       0         0      0       0     ];
%10

% Constructing F
ab0 = [a0; b0];

L12_u = L12*u;

F0 = [G - b0; a0 - L11*b0 - L12_u];

flag = true;
iter = 1;
alpha=1;
while(flag)

    temp = NE.AACC*(DG/(NE.eye_na+NE.AA*DG)); % (na+nb) x na
    J_F0=[NE.eye_J(:,1:S.na)-temp,NE.eye_J(:,S.na+1:end)]*(NE.UJ0inv*(NE.LJ0inv*F0));

    ab = ab0 - alpha*(J_F0);
    anew = ab(NE.indexa);

    % G
    PB0 = (0.0000050*(anew(8))^3 - 0.0048*(anew(8))^2 + 1.54*(anew(8)) - 164.9);
    PC0 = (0.00001038*(anew(8))^3 - 0.0098*(anew(8))^2 + 3.14*(anew(8)) -339.75);
```

```
    G = [   x0(1) + x0(2) + x0(3)                                          %=
b(1) = ml     = g1(x,a)
        (C.Pi/C.Tis)*anew(13);
%= b(2) = Phe    = g2(x,a)
        (-0.069 * sqrt(C.rhovi) * ((C.Pi - anew(12))^3 - 36 * (C.Pi - anew(12))^2 - 69 *
(C.Pi - anew(12)))/((C.Pi - anew(12))+0.08)) * anew(9); %= b(3) = mdothe = g3(x,a)
        (C.Cpvi*C.Tis - C.Cpli*anew(13) + C.hvi)*anew(3);
%= b(4) = Q      = g4(x,a)
        x0(1)/(anew(11) + C.ep);
%= b(5) = wA    = g5(x,a)
        x0(2)/(anew(11) + C.ep);
%= b(6) = wB    = g6(x,a)
        x0(3)/(anew(11) + C.ep);
%= b(7) = wC    = g7(x,a)
        x0(4)/((anew(11) + C.ep)*(3.05*anew(5) + 4.315*anew(6) + 3.5*anew(7)));
%= b(8) = T     = g8(x,a)
        ((anew(11)/(1046.085*anew(5) + 930.42*anew(6) + 658.9*anew(7))) - C.VB)/C.vL;
%= b(9) = L     = g9(x,a)
        ((anew(6)/C.MB)*PB0 +
(anew(7)/C.MC)*PC0)/((anew(5)/C.MA)+(anew(6)/C.MB)+(anew(7)/C.MC))]; %= b(10)= P     =
g10(x,a)

    % DG
    dg2_a13 = (C.Pi/C.Tis);
    dg3_a9 = (-0.069 * sqrt(C.rhovi) * ((C.Pi - anew(12))^3 - 36 * (C.Pi - anew(12))^2 -
69 * (C.Pi - anew(12)))/((C.Pi - anew(12))+0.08));
    dg3_a12 = (69*C.rhovi^(1/2)*anew(9)*(69*C.Pi - 69*anew(12) + 36*(C.Pi - anew(12))^2 -
(C.Pi - anew(12))^3))/(1000*(C.Pi - anew(12) + 2/25)^2) -
(69*C.rhovi^(1/2)*anew(9)*(72*C.Pi - 72*anew(12) - 3*(C.Pi - anew(12))^2 +
69))/(1000*(C.Pi - anew(12) + 2/25));
    dg4_a3 = (C.Cpvi*C.Tis - C.Cpli*anew(13) + C.hvi);
    dg4_a13 = -C.Cpli*anew(3);
    dg5_a11 = -x0(1)/((anew(11) + C.ep)^2);
    dg6_a11 = -x0(2)/((anew(11) + C.ep)^2);
    dg7_a11 = -x0(3)/((anew(11) + C.ep)^2);
    dg8_a5 = -(61*x0(4))/(20*(C.ep + anew(11))*((61*anew(5))/20 + (863*anew(6))/200 +
(7*anew(7))/2)^2);
    dg8_a6 = -(863*x0(4))/(200*(C.ep + anew(11))*((61*anew(5))/20 + (863*anew(6))/200 +
(7*anew(7))/2)^2);
    dg8_a7 = -(7*x0(4))/(2*(C.ep + anew(11))*((61*anew(5))/20 + (863*anew(6))/200 +
(7*anew(7))/2)^2);
    dg8_a11 = -x0(4)/(((anew(11) + C.ep)^2)*(3.05*anew(5) + 4.315*anew(6) +
3.5*anew(7)));
    dg9_a5 = -
(1150182621142057*anew(11))/(1099511627776*C.vL*((1150182621142057*anew(5))/1099511627776
+ (46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg9_a6 = -(46521*anew(11))/(50*C.vL*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg9_a7 = -(6589*anew(11))/(10*C.vL*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg9_a11 = 1/(C.vL*((1150182621142057*anew(5))/1099511627776 + (46521*anew(6))/50 +
(6589*anew(7))/10));
```

```
    dg10_a5 = -((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MA*(anew(5)/C.MA +
anew(6)/C.MB + anew(7)/C.MC)^2);
    dg10_a6 = PB0/(C.MB*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MB*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg10_a7 = PC0/(C.MC*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MC*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg10_a8 = ((anew(6)*((17708874310761171*anew(8)^2)/1180591620717411303424 -
(6*anew(8))/625 + 77/50))/C.MB +
(anew(7)*((18381811534570095*anew(8)^2)/590295810358705651712 - (49*anew(8))/2500 +
157/50))/C.MC)/(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC);

    %       1 2    3    4     5      6      7       8      9      10       11     12
13
    DG = [  0 0    0    0     0      0      0       0      0      0        0      0       0
%1
        0 0    0    0     0      0      0       0      0      0        0      0    dg2_a13
%2
        0 0    0    0     0      0      0       0    dg3_a9   0        0  dg3_a12    0
%3
        0 0 dg4_a3  0     0      0      0       0      0      0        0      0    dg4_a13
%4
        0 0    0    0     0      0      0       0      0      0     dg5_a11   0       0
%5
        0 0    0    0     0      0      0       0      0      0     dg6_a11   0       0
%6
        0 0    0    0     0      0      0       0      0      0     dg7_a11   0       0
%7
        0 0    0    0  dg8_a5 dg8_a6 dg8_a7     0      0      0     dg8_a11   0       0
%8
        0 0    0    0  dg9_a5 dg9_a6 dg9_a7     0      0      0     dg9_a11   0       0
%9
        0 0    0    0 dg10_a5 dg10_a6 dg10_a7 dg10_a8  0      0        0      0       0
]; %10

    F = [G - ab(NE.indexb); ab(NE.indexa) - L11*ab(NE.indexb) - L12_u];

    alpha=1;
    if(norm(F,inf) < 0.01) % Pass this or global options.Ftol
        flag = false;
    else
        ab0 = ab;
        F0 = F;
        iter = iter+1;
    end
end
a = ab(NE.indexa);
b = ab(NE.indexb);
% iter
end
```

```
function [a,b] = NRE(x0,u,v,a0,b0,S,C,E)

% Newton Raphson Algorithm for solving for a,b
%{
    Initial guesses are: a0 , b0
    Variables Passed from Relaxation Algo: x0, u, v

    v takes values in {0,1}

    Options:
    Tolerances: Xtol (default (1e-6)), Ftol (default (1e-6))
%}
% Calling the Gfun to obtain G and DG
L11 = S.L11;
L12 = S.L12;

% Find Value of G
PB0 = (0.0000050*(a0(8))^3 - 0.0048*(a0(8))^2 + 1.54*(a0(8)) - 164.9);
PC0 = (0.00001038*(a0(8))^3 - 0.0098*(a0(8))^2 + 3.14*(a0(8)) -339.75);

G = [   x0(1) + x0(2) + x0(3) - a0(12)*a0(13)                                    %b(1) =
ml      = g1(x,a)
    (a0(6)*PB0 + a0(7)*PC0)/(C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC))  %b(2) =
rhov    = g2(x,a)
    C.V - a0(11)/(1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7))                  %b(3) =
Vv      = g3(x,a)
    (C.Cpvi*C.Tis - C.Cpli*a0(16) + C.hvi)*a0(4)                                %b(4) = Q
= g4(x,a)
    x0(1)/(a0(11) + C.ep)                                                       %b(5) =
wA      = g5(x,a)
    (x0(2)-a0(9)*a0(12)*a0(13))/(a0(11) + C.ep)                                 %b(6) =
wB      = g6(x,a)   = (C.Cpvi*C.Tis - Cpli*a(7) + hvi)*a(4)
    (x0(3)-a0(10)*a0(12)*a0(13))/(a0(11) + C.ep)                                %b(7) =
wC      = g7(x,a)   =
    (x0(4)-(2375.8752*a0(9) + 1184.8742*a0(10))*a0(12)*a0(13))/((a0(11) +
C.ep)*(3.05*a0(5) + 4.315*a0(6) + 3.5*a0(7)) + a0(12)*a0(13)*(0.0019325*a0(9) +
0.0016115*a0(10)))               %b(8) = T       = g8(x,a)
    a0(6)*PB0/(a0(6)*PB0 + a0(7)*PC0)                                           %b(9) =
EB      = g9(x,a)   = ((x(2)/(a(3)*C.MB))*F.PB0 +
(x(3)/(a(3)*C.MC))*F.PC0)/((x(1)/(a(3)*C.MA))+(x(2)/(a(3)*C.MB))+(x(3)/(a(3)*C.MC)))
    a0(7)*PC0/(a0(6)*PB0 + a0(7)*PC0)                                           %b(10)=
EC      = g10(x,a) = ((a(3)/((1046.085*x(1) + 930.42*x(2) + 658.9*x(3))/(a(3)+ep))) -
C.VB)/C.vL
    ((a0(11)/(1046.085*a0(5) + 930.42*a0(6) + 658.9*a0(7)))-C.VB)/C.vL          %b(11)= L
= g11(x,a)
    ((a0(6)/C.MB)*PB0 + (a0(7)/C.MC)*PC0)/(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)%b(12)= P
= g12(x,a)
    (-0.069 * sqrt(a0(12)) * ((a0(18) - C.Pa)^3 - 36 * (a0(18) - C.Pa)^2 - 69 * (a0(18) -
C.Pa))/((a0(18) - C.Pa)+0.08)) * a0(19)    %b(13)= mdot_ov = g13(x,a)
    (-0.069 * sqrt(C.rhovi) * ((C.Pi - a0(17))^3 - 36 * (C.Pi - a0(17))^2 - 69 * (C.Pi -
a0(17)))/((C.Pi - a0(17))+0.08)) * a0(14)   %b(14)= mhot_he = g14(x,a)
```

```
    (C.Pi/C.Tis)*a0(16)                                                    ];%b(15)=
Phe     = g15(x,a)

% DG
dg2_a5  = -(PB0*a0(6) + PC0*a0(7))/(C.MA*C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB +
a0(7)/C.MC)^2);
dg2_a6  = PB0/(C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - (PB0*a0(6) +
PC0*a0(7))/(C.MB*C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg2_a7  = PC0/(C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - (PB0*a0(6) +
PC0*a0(7))/(C.MC*C.R*a0(8)*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg2_a8  = -(PB0*a0(6) + PC0*a0(7))/(C.R*a0(8)^2*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC));
dg3_a5  =
(1150182621142057*a0(11))/(1099511627776*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg3_a6  =
(1150182621142057*a0(11))/(1099511627776*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg3_a7  =
(1150182621142057*a0(11))/(1099511627776*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg3_a11 =
(1150182621142057*a0(11))/(1099511627776*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg4_a4  = (C.Cpvi*C.Tis - C.Cpli*a0(16) + C.hvi);
dg4_a16 = - C.Cpli*a0(4);
dg5_a11 = -x0(1)/((a0(11) + C.ep)^2);
dg6_a9  = -(a0(12)*a0(13))/(a0(11) + C.ep);
dg6_a11 = -(x0(2)-a0(9)*a0(12)*a0(13))/((a0(11) + C.ep)^2);
dg6_a12 = -(a0(9)*a0(13))/(a0(11) + C.ep);
dg6_a13 = -(a0(9)*a0(12))/(a0(11) + C.ep);
dg7_a10 = -(a0(12)*a0(13))/(a0(11) + C.ep);
dg7_a11 = -(x0(3)-a0(10)*a0(12)*a0(13))/((a0(11) + C.ep)^2);
dg7_a12 = -(a0(10)*a0(13))/(a0(11) + C.ep);
dg7_a13 = -(a0(10)*a0(12))/(a0(11) + C.ep);
dg8_a5  = -((x0(4) - a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((61*C.ep)/20 + (61*a0(11))/20))/((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg8_a6  = -((x0(4) - a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((863*C.ep)/200 + (863*a0(11))/200))/((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg8_a7  = -((x0(4) - a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((7*C.ep)/2 + (7*a0(11))/2))/((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg8_a9  = - (5224604817089259*a0(12)*a0(13))/(2199023255552*((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
```

```
(928966502336967*a0(10))/576460752303423488))) - (8912083230610927*a0(12)*a0(13)*(x0(4) -
a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104)))/(4611686018427387904*((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2);
dg8_a10 = - (5211131841407143*a0(12)*a0(13))/(4398046511104*((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))) - (928966502336967*a0(12)*a0(13)*(x0(4) -
a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104)))/(576460752303423488*((C.ep +
a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2);
dg8_a11 = -((x0(4) - a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((61*a0(5))/20 + (863*a0(6))/200 +
(7*a0(7))/2))/((C.ep + a0(11))*((61*a0(5))/20 + (863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg8_a12 = - (a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))/((C.ep + a0(11))*((61*a0(5))/20 +
(863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488)) - (a0(13)*(x0(4) -
a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))/((C.ep + a0(11))*((61*a0(5))/20 +
(863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg8_a13 = - (a0(12)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))/((C.ep + a0(11))*((61*a0(5))/20 +
(863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488)) - (a0(12)*(x0(4) -
a0(12)*a0(13)*((5224604817089259*a0(9))/2199023255552 +
(5211131841407143*a0(10))/4398046511104))*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))/((C.ep + a0(11))*((61*a0(5))/20 +
(863*a0(6))/200 + (7*a0(7))/2) +
a0(12)*a0(13)*((8912083230610927*a0(9))/4611686018427387904 +
(928966502336967*a0(10))/576460752303423488))^2;
dg9_a6  = PB0/(PB0*a0(6) + PC0*a0(7)) - (PB0^2*a0(6))/(PB0*a0(6) + PC0*a0(7))^2;
dg9_a7  = -(PB0*PC0*a0(6))/(PB0*a0(6) + PC0*a0(7))^2;
dg9_a8  = (a0(6)*((17708874310761171*a0(8)^2)/1180591620717411303424 - (6*a0(8))/625 +
77/50))/(a0(7)*((6127270511523365*a0(8)^3)/590295810358705651712 - (49*a0(8)^2)/5000 +
(157*a0(8))/50 - 1359/4) + a0(6)*((5902958103587057*a0(8)^3)/1180591620717411303424 -
(3*a0(8)^2)/625 + (77*a0(8))/50 - 1649/10)) -
(a0(6)*(a0(7)*((18381811534570095*a0(8)^2)/590295810358705651712 - (49*a0(8))/2500 +
157/50) + a0(6)*((17708874310761171*a0(8)^2)/1180591620717411303424 - (6*a0(8))/625 +
77/50))*((5902958103587057*a0(8)^3)/1180591620717411303424 - (3*a0(8)^2)/625 +
(77*a0(8))/50 - 1649/10))/(a0(7)*((6127270511523365*a0(8)^3)/590295810358705651712 -
```

```
(49*a0(8)^2)/5000 + (157*a0(8))/50 - 1359/4) +
a0(6)*((5902958103587057*a0(8)^3)/1180591620717411303424 - (3*a0(8)^2)/625 +
(77*a0(8))/50 - 1649/10))^2;
dg10_a6 = -(PB0*PC0*a0(7))/(PB0*a0(6) + PC0*a0(7))^2;
dg10_a7 = PC0/(PB0*a0(6) + PC0*a0(7)) - (PC0^2*a0(7))/(PB0*a0(6) + PC0*a0(7))^2;
dg10_a8 = (a0(7)*((18381811534570095*a0(8)^2)/590295810358705651712 - (49*a0(8))/2500 +
157/50))/(a0(7)*((6127270511523365*a0(8)^3)/590295810358705651712 - (49*a0(8)^2)/5000 +
(157*a0(8))/50 - 1359/4) + a0(6)*((5902958103587057*a0(8)^3)/1180591620717411303424 -
(3*a0(8)^2)/625 + (77*a0(8))/50 - 1649/10)) -
(a0(7)*(a0(7)*((18381811534570095*a0(8)^2)/590295810358705651712 - (49*a0(8))/2500 +
157/50) + a0(6)*((17708874310761171*a0(8)^2)/1180591620717411303424 - (6*a0(8))/625 +
77/50))*((6127270511523365*a0(8)^3)/590295810358705651712 - (49*a0(8)^2)/5000 +
(157*a0(8))/50 - 1359/4))/(a0(7)*((6127270511523365*a0(8)^3)/590295810358705651712 -
(49*a0(8)^2)/5000 + (157*a0(8))/50 - 1359/4) +
a0(6)*((5902958103587057*a0(8)^3)/1180591620717411303424 - (3*a0(8)^2)/625 +
(77*a0(8))/50 - 1649/10))^2;
dg11_a5 = -
(1150182621142057*a0(11))/(1099511627776*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg11_a6 = -(46521*a0(11))/(50*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg11_a7 = -(6589*a0(11))/(10*C.vL*((1150182621142057*a0(5))/1099511627776 +
(46521*a0(6))/50 + (6589*a0(7))/10)^2);
dg11_a11= 1/(C.vL*((1150182621142057*a0(5))/1099511627776 + (46521*a0(6))/50 +
(6589*a0(7))/10));
dg12_a5 = -((PB0*a0(6))/C.MB + (PC0*a0(7))/C.MC)/(C.MA*(a0(5)/C.MA + a0(6)/C.MB +
a0(7)/C.MC)^2);
dg12_a6 = PB0/(C.MB*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - ((PB0*a0(6))/C.MB +
(PC0*a0(7))/C.MC)/(C.MB*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg12_a7 = PC0/(C.MC*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)) - ((PB0*a0(6))/C.MB +
(PC0*a0(7))/C.MC)/(C.MC*(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC)^2);
dg12_a8 = ((a0(6)*((17708874310761171*a0(8)^2)/1180591620717411303424 - (6*a0(8))/625 +
77/50))/C.MB + (a0(7)*((18381811534570095*a0(8)^2)/590295810358705651712 -
(49*a0(8))/2500 + 157/50))/C.MC)/(a0(5)/C.MA + a0(6)/C.MB + a0(7)/C.MC);
dg13_a12= (69*a0(19)*(69*a0(18) - 69*C.Pa + 36*(C.Pa - a0(18))^2 + (C.Pa -
a0(18))^3))/(2000*a0(12)^(1/2)*(a0(18) - C.Pa + 2/25));
dg13_a18= -(69*a0(12)^(1/2)*a0(19)*(72*C.Pa - 72*a0(18) + 3*(C.Pa - a0(18))^2 -
69))/(1000*(a0(18) - C.Pa + 2/25)) - (69*a0(12)^(1/2)*a0(19)*(69*a0(18) - 69*C.Pa +
36*(C.Pa - a0(18))^2 + (C.Pa - a0(18))^3))/(1000*(a0(18) - C.Pa + 2/25)^2);
dg13_a19= (69*a0(12)^(1/2)*(69*a0(18) - 69*C.Pa + 36*(C.Pa - a0(18))^2 + (C.Pa -
a0(18))^3))/(1000*(a0(18) - C.Pa + 2/25));
dg14_a14= (69*C.rhovi^(1/2)*(69*C.Pi - 69*a0(17) + 36*(C.Pi - a0(17))^2 - (C.Pi -
a0(17))^3))/(1000*(C.Pi - a0(17) + 2/25));
dg14_a17= (69*C.rhovi^(1/2)*a0(14)*(69*C.Pi - 69*a0(17) + 36*(C.Pi - a0(17))^2 - (C.Pi -
a0(17))^3))/(1000*(C.Pi - a0(17) + 2/25)^2) - (69*C.rhovi^(1/2)*a0(14)*(72*C.Pi -
72*a0(17) - 3*(C.Pi - a0(17))^2 + 69))/(1000*(C.Pi - a0(17) + 2/25));

%   1 2 3  4     5      6      7       8      9      10       11     12     13      14
15   16     17    18       19
DG = [  0 0 0  0     0      0      0       0      0      0        0   -a0(13) -a0(12) 0
0    0         0     0       0       %1
    0 0 0  0   dg2_a5 dg2_a6 dg2_a7 dg2_a8    0      0        0      0      0      0    0
```

```
0         0      0        0        %2
    0 0 0  0   dg3_a5 dg3_a6 dg3_a7    0       0       0    dg3_a11   0       0       0    0
0         0      0        0        %3
    0 0 0 dg4_a4 0       0       0        0       0       0        0      0       0       0    0
dg4_a16     0      0        0        %4
    0 0 0  0      0       0       0        0       0       0    dg5_a11   0       0       0    0
0         0      0        0        %5
    0 0 0  0      0       0       0        0   dg6_a9    0    dg6_a11 dg6_a12 dg6_a13 0    0
0         0      0        0        %6
    0 0 0  0      0       0       0        0       0   dg7_a10 dg7_a11 dg7_a12 dg7_a13 0    0
0         0      0        0        %7
    0 0 0  0   dg8_a5   dg8_a6 dg8_a7    0   dg8_a9 dg8_a10 dg8_a11 dg8_a12 dg8_a13 0    0
0         0      0        0        %8
    0 0 0  0      0     dg9_a6 dg9_a7 dg9_a8    0       0        0      0       0       0    0
0         0      0        0        %9
    0 0 0  0      0  dg10_a6   dg10_a7 dg10_a8  0       0        0      0       0       0    0
0         0      0        0       %10
    0 0 0  0 dg11_a5 dg11_a6  dg11_a7   0       0       0   dg11_a11   0       0       0    0
0         0      0        0       %11
    0 0 0  0  dg12_a5 dg12_a6 dg12_a7 dg12_a8 0       0        0      0       0       0    0
0         0      0        0       %12
    0 0 0  0      0       0       0        0       0       0        0  dg13_a12  0       0    0
0         0  dg13_a18 dg13_a19 %13
    0 0 0  0      0       0       0        0       0       0        0      0       0  dg14_a14 0
0    dg14_a17  0        0       %14
    0 0 0  0      0       0       0        0       0       0        0      0       0       0    0
(C.Pi/C.Tis) 0     0        0       %15
    ];

% Constructing F
ab0 = [a0; b0];

L12_u = L12*u;

F0 = [G - b0; a0 - L11*b0 - L12_u];

flag = true;
iter = 1;
alpha=1;
while(flag)

    temp = E.AACC*(DG/(E.eye_na+E.AA*DG)); % (na+nb) x na
    J_F0=[E.eye_J(:,1:S.na)-temp,E.eye_J(:,S.na+1:end)]*(E.UJ0inv*(E.LJ0inv*F0));

    ab = ab0 - alpha*(J_F0);
    anew = ab(E.indexa);

    % Find Value of G
    PB0 = (0.0000050*(anew(8))^3 - 0.0048*(anew(8))^2 + 1.54*(anew(8)) - 164.9);
    PC0 = (0.00001038*(anew(8))^3 - 0.0098*(anew(8))^2 + 3.14*(anew(8)) -339.75);

    G = [   x0(1) + x0(2) + x0(3) - anew(12)*anew(13)
```

```
%b(1) = ml       = g1(x,a)
        (anew(6)*PB0 + anew(7)*PC0)/(C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC))  %b(2) = rhov    = g2(x,a)
        C.V - anew(11)/(1046.085*anew(5) + 930.42*anew(6) + 658.9*anew(7))
%b(3) = Vv       = g3(x,a)
        (C.Cpvi*C.Tis - C.Cpli*anew(16) + C.hvi)*anew(4)
%b(4) = Q        = g4(x,a)
        x0(1)/(anew(11) + C.ep)
%b(5) = wA       = g5(x,a)
        (x0(2)-anew(9)*anew(12)*anew(13))/(anew(11) + C.ep)
%b(6) = wB       = g6(x,a)   = (C.Cpvi*C.Tis - Cpli*a(7) + hvi)*a(4)
        (x0(3)-anew(10)*anew(12)*anew(13))/(anew(11) + C.ep)
%b(7) = wC       = g7(x,a)   =
        (x0(4)-(2375.8752*anew(9) + 1184.8742*anew(10))*anew(12)*anew(13))/((anew(11) +
C.ep)*(3.05*anew(5) + 4.315*anew(6) + 3.5*anew(7)) + anew(12)*anew(13)*(0.0019325*anew(9)
+ 0.0016115*anew(10)))              %b(8) = T       = g8(x,a)
        anew(6)*PB0/(anew(6)*PB0 + anew(7)*PC0)
%b(9) = EB       = g9(x,a)   = ((x(2)/(a(3)*C.MB))*F.PB0 +
(x(3)/(a(3)*C.MC))*F.PC0)/((x(1)/(a(3)*C.MA))+(x(2)/(a(3)*C.MB))+(x(3)/(a(3)*C.MC)))
        anew(7)*PC0/(anew(6)*PB0 + anew(7)*PC0)
%b(10)= EC       = g10(x,a) = ((a(3)/((1046.085*x(1) + 930.42*x(2) +
658.9*x(3))/(a(3)+ep))) - C.VB)/C.vL
        ((anew(11)/(1046.085*anew(5) + 930.42*anew(6) + 658.9*anew(7)))-C.VB)/C.vL
%b(11)= L        = g11(x,a)
        ((anew(6)/C.MB)*PB0 + (anew(7)/C.MC)*PC0)/(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)%b(12)= P       = g12(x,a)
        (-0.069 * sqrt(anew(12)) * ((anew(18) - C.Pa)^3 - 36 * (anew(18) - C.Pa)^2 - 69 *
(anew(18) - C.Pa))/((anew(18) - C.Pa)+0.08)) * anew(19)    %b(13)= mdot_ov = g13(x,a)
        (-0.069 * sqrt(C.rhovi) * ((C.Pi - anew(17))^3 - 36 * (C.Pi - anew(17))^2 - 69 *
(C.Pi - anew(17)))/((C.Pi - anew(17))+0.08)) * anew(14)   %b(14)= mhot_he = g14(x,a)
        (C.Pi/C.Tis)*anew(16)
];%b(15)= Phe      = g15(x,a)

    % DG
    dg2_a5   = -(PB0*anew(6) + PC0*anew(7))/(C.MA*C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB
+ anew(7)/C.MC)^2);
    dg2_a6   = PB0/(C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
(PB0*anew(6) + PC0*anew(7))/(C.MB*C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg2_a7   = PC0/(C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
(PB0*anew(6) + PC0*anew(7))/(C.MC*C.R*anew(8)*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg2_a8   = -(PB0*anew(6) + PC0*anew(7))/(C.R*anew(8)^2*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC));
    dg3_a5   =
(1150182621142057*anew(11))/(1099511627776*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg3_a6   =
(1150182621142057*anew(11))/(1099511627776*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg3_a7   =
(1150182621142057*anew(11))/(1099511627776*((1150182621142057*anew(5))/1099511627776 +
```

```
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg3_a11 =
(1150182621142057*anew(11))/(1099511627776*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg4_a4  = (C.Cpvi*C.Tis - C.Cpli*anew(16) + C.hvi);
    dg4_a16 = - C.Cpli*anew(4);
    dg5_a11 = -x0(1)/((anew(11) + C.ep)^2);
    dg6_a9  = -(anew(12)*anew(13))/(anew(11) + C.ep);
    dg6_a11 = -(x0(2)-anew(9)*anew(12)*anew(13))/((anew(11) + C.ep)^2);
    dg6_a12 = -(anew(9)*anew(13))/(anew(11) + C.ep);
    dg6_a13 = -(anew(9)*anew(12))/(anew(11) + C.ep);
    dg7_a10 = -(anew(12)*anew(13))/(anew(11) + C.ep);
    dg7_a11 = -(x0(3)-anew(10)*anew(12)*anew(13))/((anew(11) + C.ep)^2);
    dg7_a12 = -(anew(10)*anew(13))/(anew(11) + C.ep);
    dg7_a13 = -(anew(10)*anew(12))/(anew(11) + C.ep);
    dg8_a5  = -((x0(4) - anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((61*C.ep)/20 + (61*anew(11))/20))/((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg8_a6  = -((x0(4) - anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((863*C.ep)/200 + (863*anew(11))/200))/((C.ep
+ anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg8_a7  = -((x0(4) - anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((7*C.ep)/2 + (7*anew(11))/2))/((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg8_a9  = - (5224604817089259*anew(12)*anew(13))/(2199023255552*((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))) -
(8912083230610927*anew(12)*anew(13)*(x0(4) -
anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104)))/(4611686018427387904*((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2);
    dg8_a10 = - (5211131841407143*anew(12)*anew(13))/(4398046511104*((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))) -
(928966502336967*anew(12)*anew(13)*(x0(4) -
anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104)))/(576460752303423488*((C.ep +
anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2);
    dg8_a11 = -((x0(4) - anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((61*anew(5))/20 + (863*anew(6))/200 +
```

```
(7*anew(7))/2))/((C.ep + anew(11))*((61*anew(5))/20 + (863*anew(6))/200 + (7*anew(7))/2)
+ anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg8_a12 = - (anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))/((C.ep + anew(11))*((61*anew(5))/20 +
(863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488)) - (anew(13)*(x0(4) -
anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((8912083230610927*anew(9))/46116860184273879
04 + (928966502336967*anew(10))/576460752303423488))/((C.ep + anew(11))*((61*anew(5))/20
+ (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg8_a13 = - (anew(12)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))/((C.ep + anew(11))*((61*anew(5))/20 +
(863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488)) - (anew(12)*(x0(4) -
anew(12)*anew(13)*((5224604817089259*anew(9))/2199023255552 +
(5211131841407143*anew(10))/4398046511104))*((8912083230610927*anew(9))/46116860184273879
04 + (928966502336967*anew(10))/576460752303423488))/((C.ep + anew(11))*((61*anew(5))/20
+ (863*anew(6))/200 + (7*anew(7))/2) +
anew(12)*anew(13)*((8912083230610927*anew(9))/4611686018427387904 +
(928966502336967*anew(10))/576460752303423488))^2;
    dg9_a6  = PB0/(PB0*anew(6) + PC0*anew(7)) - (PB0^2*anew(6))/(PB0*anew(6) +
PC0*anew(7))^2;
    dg9_a7  = -(PB0*PC0*anew(6))/(PB0*anew(6) + PC0*anew(7))^2;
    dg9_a8  = (anew(6)*((17708874310761171*anew(8)^2)/1180591620717411303424 -
(6*anew(8))/625 + 77/50))/(anew(7)*((6127270511523365*anew(8)^3)/590295810358705651712 -
(49*anew(8)^2)/5000 + (157*anew(8))/50 - 1359/4) +
anew(6)*((5902958103587057*anew(8)^3)/1180591620717411303424 - (3*anew(8)^2)/625 +
(77*anew(8))/50 - 1649/10)) -
(anew(6)*(anew(7)*((18381811534570095*anew(8)^2)/590295810358705651712 -
(49*anew(8))/2500 + 157/50) +
anew(6)*((17708874310761171*anew(8)^2)/1180591620717411303424 - (6*anew(8))/625 +
77/50))*((5902958103587057*anew(8)^3)/1180591620717411303424 - (3*anew(8)^2)/625 +
(77*anew(8))/50 - 1649/10))/(anew(7)*((6127270511523365*anew(8)^3)/590295810358705651712
- (49*anew(8)^2)/5000 + (157*anew(8))/50 - 1359/4) +
anew(6)*((5902958103587057*anew(8)^3)/1180591620717411303424 - (3*anew(8)^2)/625 +
(77*anew(8))/50 - 1649/10))^2;
    dg10_a6 = -(PB0*PC0*anew(7))/(PB0*anew(6) + PC0*anew(7))^2;
    dg10_a7 = PC0/(PB0*anew(6) + PC0*anew(7)) - (PC0^2*anew(7))/(PB0*anew(6) +
PC0*anew(7))^2;
    dg10_a8 = (anew(7)*((18381811534570095*anew(8)^2)/590295810358705651712 -
(49*anew(8))/2500 + 157/50))/(anew(7)*((6127270511523365*anew(8)^3)/590295810358705651712
- (49*anew(8)^2)/5000 + (157*anew(8))/50 - 1359/4) +
anew(6)*((5902958103587057*anew(8)^3)/1180591620717411303424 - (3*anew(8)^2)/625 +
(77*anew(8))/50 - 1649/10)) -
(anew(7)*(anew(7)*((18381811534570095*anew(8)^2)/590295810358705651712 -
(49*anew(8))/2500 + 157/50) +
anew(6)*((17708874310761171*anew(8)^2)/1180591620717411303424 - (6*anew(8))/625 +
```

```
77/50))*((6127270511523365*anew(8)^3)/590295810358705651712 - (49*anew(8)^2)/5000 +
(157*anew(8))/50 - 1359/4))/(anew(7)*((6127270511523365*anew(8)^3)/590295810358705651712
- (49*anew(8)^2)/5000 + (157*anew(8))/50 - 1359/4) +
anew(6)*((5902958103587057*anew(8)^3)/1180591620717411303424 - (3*anew(8)^2)/625 +
(77*anew(8))/50 - 1649/10))^2;
    dg11_a5 = -
(1150182621142057*anew(11))/(1099511627776*C.vL*((1150182621142057*anew(5))/1099511627776
+ (46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg11_a6 = -(46521*anew(11))/(50*C.vL*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg11_a7 = -(6589*anew(11))/(10*C.vL*((1150182621142057*anew(5))/1099511627776 +
(46521*anew(6))/50 + (6589*anew(7))/10)^2);
    dg11_a11= 1/(C.vL*((1150182621142057*anew(5))/1099511627776 + (46521*anew(6))/50 +
(6589*anew(7))/10));
    dg12_a5 = -((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MA*(anew(5)/C.MA +
anew(6)/C.MB + anew(7)/C.MC)^2);
    dg12_a6 = PB0/(C.MB*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MB*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg12_a7 = PC0/(C.MC*(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC)) -
((PB0*anew(6))/C.MB + (PC0*anew(7))/C.MC)/(C.MC*(anew(5)/C.MA + anew(6)/C.MB +
anew(7)/C.MC)^2);
    dg12_a8 = ((anew(6)*((17708874310761171*anew(8)^2)/1180591620717411303424 -
(6*anew(8))/625 + 77/50))/C.MB +
(anew(7)*((18381811534570095*anew(8)^2)/590295810358705651712 - (49*anew(8))/2500 +
157/50))/C.MC)/(anew(5)/C.MA + anew(6)/C.MB + anew(7)/C.MC);
    dg13_a12= (69*anew(19)*(69*anew(18) - 69*C.Pa + 36*(C.Pa - anew(18))^2 + (C.Pa -
anew(18))^3))/(2000*anew(12)^(1/2)*(anew(18) - C.Pa + 2/25));
    dg13_a18= -(69*anew(12)^(1/2)*anew(19)*(72*C.Pa - 72*anew(18) + 3*(C.Pa - anew(18))^2
- 69))/(1000*(anew(18) - C.Pa + 2/25)) - (69*anew(12)^(1/2)*anew(19)*(69*anew(18) -
69*C.Pa + 36*(C.Pa - anew(18))^2 + (C.Pa - anew(18))^3))/(1000*(anew(18) - C.Pa +
2/25)^2);
    dg13_a19= (69*anew(12)^(1/2)*(69*anew(18) - 69*C.Pa + 36*(C.Pa - anew(18))^2 + (C.Pa
- anew(18))^3))/(1000*(anew(18) - C.Pa + 2/25));
    dg14_a14= (69*C.rhovi^(1/2)*(69*C.Pi - 69*anew(17) + 36*(C.Pi - anew(17))^2 - (C.Pi -
anew(17))^3))/(1000*(C.Pi - anew(17) + 2/25));
    dg14_a17= (69*C.rhovi^(1/2)*anew(14)*(69*C.Pi - 69*anew(17) + 36*(C.Pi - anew(17))^2
- (C.Pi - anew(17))^3))/(1000*(C.Pi - anew(17) + 2/25)^2) -
(69*C.rhovi^(1/2)*anew(14)*(72*C.Pi - 72*anew(17) - 3*(C.Pi - anew(17))^2 +
69))/(1000*(C.Pi - anew(17) + 2/25));

    %   1 2 3  4      5      6      7       8      9      10       11     12     13      14
15   16      17     18       19
    DG = [  0 0 0  0      0      0      0       0      0      0        0   -anew(13) -
anew(12) 0     0     0        0      0       0       %1
        0 0 0  0   dg2_a5 dg2_a6 dg2_a7 dg2_a8    0      0        0      0      0      0
0    0        0      0        0       %2
        0 0 0  0   dg3_a5 dg3_a6 dg3_a7     0      0      0    dg3_a11   0      0      0
0    0        0      0        0       %3
        0 0 0 dg4_a4 0      0      0       0      0      0        0      0      0      0
0 dg4_a16     0      0        0       %4
        0 0 0  0      0      0      0       0      0      0    dg5_a11   0      0      0
```

```
0     0         0      0       0       %5
        0  0  0   0      0       0       0        0    dg6_a9     0     dg6_a11 dg6_a12 dg6_a13 0
0     0         0      0       0       %6
        0  0  0   0      0       0       0        0       0    dg7_a10 dg7_a11 dg7_a12 dg7_a13 0
0     0         0      0       0       %7
        0  0  0   0    dg8_a5   dg8_a6 dg8_a7    0    dg8_a9 dg8_a10 dg8_a11 dg8_a12 dg8_a13 0
0     0         0      0       0       %8
        0  0  0   0      0      dg9_a6 dg9_a7 dg9_a8    0       0        0      0       0       0
0     0         0      0       0       %9
        0  0  0   0      0   dg10_a6   dg10_a7 dg10_a8  0       0        0      0       0       0
0     0         0      0       0      %10
        0  0  0   0 dg11_a5 dg11_a6   dg11_a7   0       0       0    dg11_a11    0       0       0
0     0         0      0       0      %11
        0  0  0   0   dg12_a5 dg12_a6 dg12_a7 dg12_a8 0       0        0      0       0       0
0     0         0      0       0      %12
        0  0  0   0      0       0       0        0       0       0        0  dg13_a12   0       0
0     0         0  dg13_a18 dg13_a19 %13
        0  0  0   0      0       0       0        0       0       0        0      0       0
dg14_a14 0      0    dg14_a17   0        0       %14
        0  0  0   0      0       0       0        0       0       0        0      0       0       0
0 (C.Pi/C.Tis) 0      0        0      %15
        ];

    F = [G - ab(E.indexb); ab(E.indexa) - L11*ab(E.indexb) - L12_u];

    alpha=1;
    if(norm(F,inf) < 0.01)
        flag = false;
    else
        ab0 = ab;
        F0 = F;
        iter = iter+1;
        if(iter>2500)
            flag = false;
            disp('iter too high. exiting NR---------------------------------------------
-------');
        end
    end
end
a = ab(E.indexa);
b = ab(E.indexb);

end
```

```
function printV(tf,t0,h_opt,V,mode)
% t0 =0;
% tf = 1200;
% h_opt = 0.1;

figure;
[time,x_ref] = x_reference(t0, tf, h_opt,V);
grid on
```

```
title('Results-new-relax');

% Plot wa, L
subplot(3,3,1), hold on, plot(time,V.wA,'r',time,x_ref(1,:),'b'),...
    xlabel('time'), ylabel('wa tracking'), grid on, legend('wa','wa-REF');
subplot(3,3,2), hold on, plot(time,V.L,'r',time,x_ref(2,:),'b'), ...
    xlabel('time'), ylabel('L tracking'), grid on, legend('L','L-REF');

% Plot u
subplot(3,3,3), hold on, stairs(time,V.V1*100,'Color','r'), ...
    xlabel('time'), ylabel('u1 feed inflow input'), grid on, legend('u1_1');
subplot(3,3,4), hold on, stairs(time,V.V2*100,'Color','r'), ...
    xlabel('time'), ylabel('u2 feed outflow input'), grid on, legend('u1_2');
subplot(3,3,5), hold on, stairs(time,V.Vv1*100,'Color','r'), ...
    xlabel('time'), ylabel('u3 vapor outflow input'), grid on, legend('u1_3');
subplot(3,3,6), hold on, stairs(time,V.Vv2*100,'Color','r'), ...
    xlabel('time'), ylabel('u4 steam valve input'), grid on, legend('u1_4');

%Plot x and mode
subplot(3,3,7), hold on,  stairs(time,V.mA,'Color','r'), ...
    stairs(time,V.mB,'Color','b'), stairs(time,V.mC,'Color','k'), ...
    xlabel('time'), ylabel('mA, mB, mC'), grid on, legend('x1','x2','x3');
subplot(3,3,8), hold on, stairs(time,V.U,'Color','r'), ...
    xlabel('time'), ylabel('x4'), grid on, legend('x4');
subplot(3,3,9), hold on, stairs(mode,'Color','r'), ...
    xlabel('time'), ylabel('mode'), grid on, legend('mode');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot V
figure;
subplot(3,3,1), hold on, plot(time,V.mil,'r'),...
    xlabel('time'), ylabel('mil'), grid on, legend('mil');
subplot(3,3,2), hold on, plot(time,V.mol,'r'), ...
    xlabel('time'), ylabel('mol'), grid on, legend('mol');
subplot(3,3,3), hold on, stairs(time,V.mov,'Color','r'), ...
    xlabel('time'), ylabel('mdot_ov'), grid on, legend('mdot_ov');
subplot(3,3,4), hold on, stairs(time,V.mhe,'Color','r'), ...
    xlabel('time'), ylabel('mdot_he'), grid on, legend('mdot_he');
subplot(3,3,5), hold on, stairs(time,V.wA,'Color','r'), ...
    xlabel('time'), ylabel('wA'), grid on, legend('wA');
subplot(3,3,6), hold on, stairs(time,V.wB,'Color','r'), ...
    xlabel('time'), ylabel('wB'), grid on, legend('wB');
subplot(3,3,7), hold on, stairs(time,V.wC,'Color','r'), ...
    xlabel('time'), ylabel('wC'), grid on, legend('wC');
subplot(3,3,8), hold on, stairs(time,V.T,'Color','r'), ...
    xlabel('time'), ylabel('T'), grid on, legend('T');
subplot(3,3,9), hold on, stairs(time,V.EB,'Color','r'), ...
    xlabel('time'), ylabel('EB'), grid on, legend('EB');

figure;
subplot(3,3,1), hold on, stairs(time,V.EC,'Color','r'), ...
    xlabel('time'), ylabel('EC'), grid on, legend('EC');
subplot(3,3,2), hold on, stairs(time,V.ml,'Color','r'), ...
```

```
    xlabel('time'), ylabel('ml'), grid on, legend('ml');
subplot(3,3,3), hold on, stairs(time,V.rhov,'Color','r'), ...
    xlabel('time'), ylabel('rhov'), grid on, legend('rhov');
subplot(3,3,4), hold on, stairs(time,V.Vv,'Color','r'), ...
    xlabel('time'), ylabel('Vv'), grid on, legend('Vv');
subplot(3,3,5), hold on, stairs(time,V.Vv2*100,'Color','r'), ...
    xlabel('time'), ylabel('u4'), grid on, legend('u4');
subplot(3,3,6), hold on, stairs(time,V.Q,'Color','r'), ...
    xlabel('time'), ylabel('Q'), grid on, legend('Q');
subplot(3,3,7), hold on, stairs(time,V.The,'Color','r'), ...
    xlabel('time'), ylabel('The'), grid on, legend('The');
subplot(3,3,8), hold on, stairs(time,V.Phe,'Color','r'), ...
    xlabel('time'), ylabel('Phe'), grid on, legend('Phe');
subplot(3,3,9), hold on, stairs(time,V.P,'Color','r'), ...
    xlabel('time'), ylabel('P'), grid on, legend('P');
end

function [tref,xref] = x_reference(t0_k, tf_k, h, V)
% construct time vector
tref = t0_k:h:tf_k;
% construct xref

if ((0.8-max(V.wA))>0.09)
    xref = repmat([0.8;100],1,numel(tref));
else
    xref = repmat([0.8;70],1,numel(tref));
end
end
```